ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

Λειτουργικά Συστήματα

Συστήματα Εισόδου/Εξόδου (Input/Output -- I/O)

I/O Hardware (Υλικό Ε/Ε)

- Το σύστημα I/O αποτελεί ένα πολύ μεγάλο τμήμα ενός Λ.Σ.
- Συνήθως "**διευθύνει" όλες τις συσκευές I/O** (π.χ., δίσκους, τερματικά, εκτυπωτές, δίκτυα, ...).
- Ένα σύστημα I/O κυρίως:
  - εκδίδει **εντολές** στις συσκευές
  - αναγνωρίζει και εξυπηρετεί τα **interrupts** των συσκευών
  - χειρίζεται **βλάβες** και **λάθη** σχετικά με I/O
- Πριν εξετάσουμε ένα I/O σύστημα από πλευράς **software**, θα δούμε πρώτα το I/O **hardware** από κοντά. Θα εστιάσουμε την προσοχή μας σε γενικές πληροφορίες για συσκευές I/O, για controllers συσκευών (device controllers), και για Direct Memory Access.

I/O Devices (Συσκευές Ε/Ε)

- Υπάρχουν πολλά και πολύ διαφορετικά είδη συσκευών Εισόδου/Εξόδου!
- Δίσκοι, τερματικά, εκτυπωτές, δίκτυα, ποντίκια, ταινίες, …

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner	400 KB/sec
Digital camcorder	4 MB/sec
52x CD-ROM	8 MB/sec
FireWire (IEEE 1394)	50 MB/sec
USB 2.0	60 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
Gigabit Ethernet	125 MB/sec
Serial ATA disk	200 MB/sec
SCSI Ultrawide 4 disk	320 MB/sec
PCI bus	528 MB/sec

I/O Devices (Συσκευές Ε/Ε)

- **Block devices**
  - Κάθε block πληροφορίας έχει **τη δική του διεύθυνση** και μπορεί να εγγραφεί ή να ανακτηθεί ανεξάρτητα από τα άλλα blocks.
  - Αποθηκεύουν πληροφορία σε **μονάδες σταθερού μεγέθους** που λέγονται **blocks**.
  - Το block αποτελεί την μικρότερη μονάδα πληροφορίας που μεταφέρεται από/προς την συσκευή.
  - Κυρίως: disks

- **Character devices**
  - Διαχειρίζονται **μη διευθυνσιοδοτούμενη πληροφορία** - δηλ. απλώς μια σειρά από chars.
  - Δεν υπάρχει η δυνατότητα τυχαίας/ανεξάρτητης προσπέλασης πληροφορίας, βάσει διεύθυνσης
  - Η πληροφορία έρχεται ως stream από bytes.
  - Κυρίως: keyboards, printers, network cards, webcam, κ.τ.λ.

- Σημειώσεις:
  - Υπάρχουν διάφορες ερμηνείες για τα χαρακτηριστικά που πρέπει να έχουν block και character devices.
  - Κάποια block devices έχουν και ένα character interface (π.χ., disks σε UNIX).
  - Όλοι συμφωνούν ότι οι δίσκοι είναι block devices και ότι τερματικά, εκτυπωτές, δίκτυα, ποντίκια είναι character devices. Πολλοί επίσης θεωρούν τις μαγνητικές ταινίες ως block devices.

Device Controllers (Ελεγκτές Συσκευών)

- Πολλές συσκευές I/O έχουν δύο διαφορετικά τμήματα: ένα **ηλεκτρονικό** και ένα **μηχανικό**. Το ηλεκτρονικό τμήμα, που είναι στην ουσία ένας επεξεργαστής, ονομάζεται **controller** ή **adapter** (**ελεγκτής**).

- Ο controller είναι μια κάρτα που μπαίνει στον υπολογιστή. Αυτή η κάρτα έχει υποδοχές για συνδέσεις με I/O συσκευές.

- Η επικοινωνία του controller με τις συσκευές γίνεται μέσω ενός **I/O bus** (**διαύλου Ε/Ε**).

- Το **I/O bus** είναι διαφορετικό από το **system bus** που συνδέει τη CPU με τη μνήμη και τους ελεγκτές (controllers).

Device Controllers (Ελεγκτές Συσκευών)

- Οι αρμοδιότητες των controllers ποικίλουν πάρα πολύ, ανάλογα με το είδος της συσκευής που διαχειρίζονται.

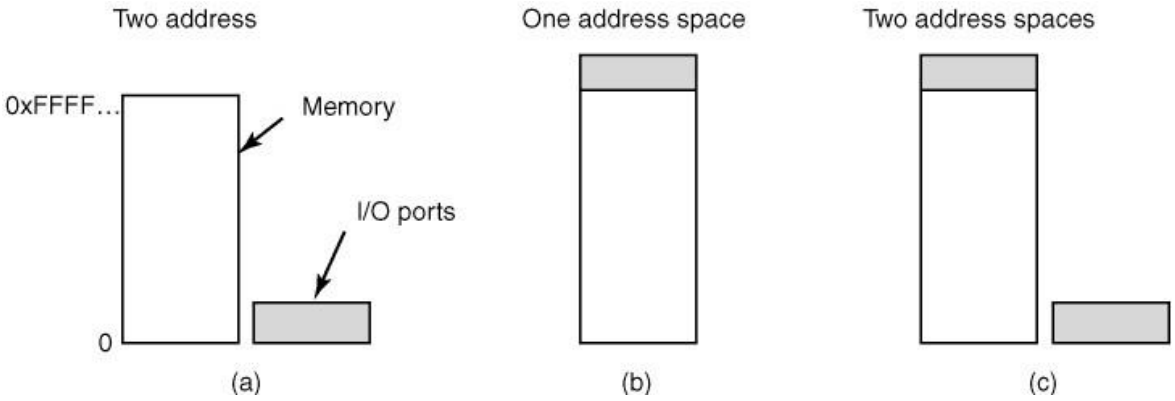
- Για παράδειγμα, μία από τις ευθύνες ενός **disk controller** είναι να κάνει έλεγχο λαθών (**error checking**).
  - Πρώτα οργανώνει τα ψηφία που έρχονται από τη συσκευή σε bytes και μετά πιστοποιεί το **checksum** που περιέχει η πληροφορία.
  - Κατόπιν, η πληροφορία μπορεί να μεταφερθεί στην Κ.Μ.

- Σαν ένα άλλο παράδειγμα, ένας **terminal controller** είναι υπεύθυνος να
  - να διαβάζει τα bytes (chars) από την Κ.Μ.
  - να καθοδηγεί την ακτίνα της CRT (οθόνης) για να εμφανιστούν οι αντίστοιχοι ASCII χαρακτήρες.
  - Επίσης, ανάλογα χειρίζεται λειτουργίες όπως scrolling, κ.τ.λ.

Επικοινωνία με Device Controllers

- Ο πιο διαδεδομένος τρόπος βασίζεται στην έννοια του **memory-mapped I/O**.
  - Ένα μέρος του χώρου διευθύνσεων (address space) του υπολογιστή αντιστοιχεί σε ειδικούς καταχωρητές, **Control and Status Registers** (**CSR**).

- Το Λ.Σ. επικοινωνεί με τον device controller χρησιμοποιώντας αυτούς τους καταχωρητές.
  - π.χ., μπορεί να "γράψει" εντολές (δίνοντας ειδικές τιμές) και να ορίσει τιμές παραμέτρων, όπως ποιο μπλοκ δίσκου πρέπει να προσπελαστεί, διεύθυνση στην Κ.Μ. όπου πρέπει να αποθηκευτεί, κ.λπ.



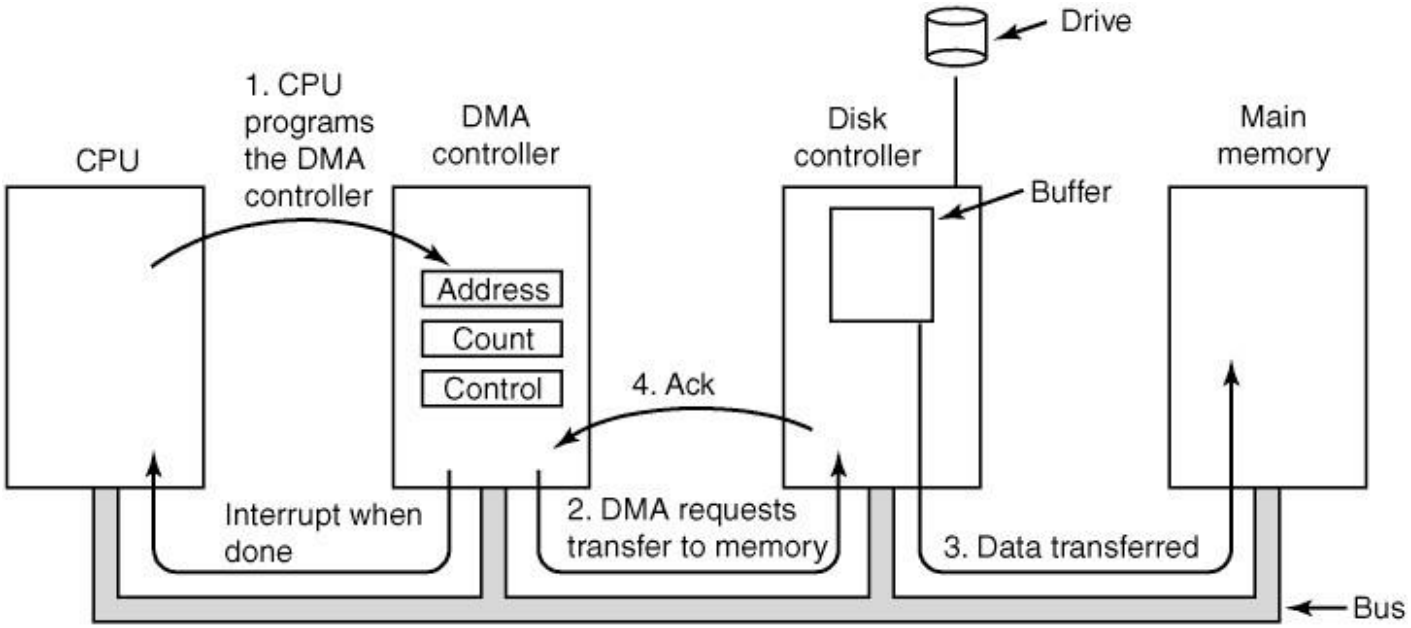
Επικοινωνία με Device Controllers

- Αφού δεχθεί μια εντολή ο device controller θα την εκτελέσει και θα γράψει πληροφορίες σχετικά με την εντολή σε ειδικούς καταχωρητές που θα εξεταστούν από το Λ.Σ.
  - π.χ., για να διαπιστωθεί αν η εντολή εκτελέστηκε επιτυχώς.

- Κατόπιν, ο ελεγκτής θα προκαλέσει μια διακοπή (interrupt) ώστε να εκτελεστεί το κατάλληλο τμήμα του Λ.Σ. και να συνεχίσει την περαιτέρω επεξεργασία της κλήσης I/O.

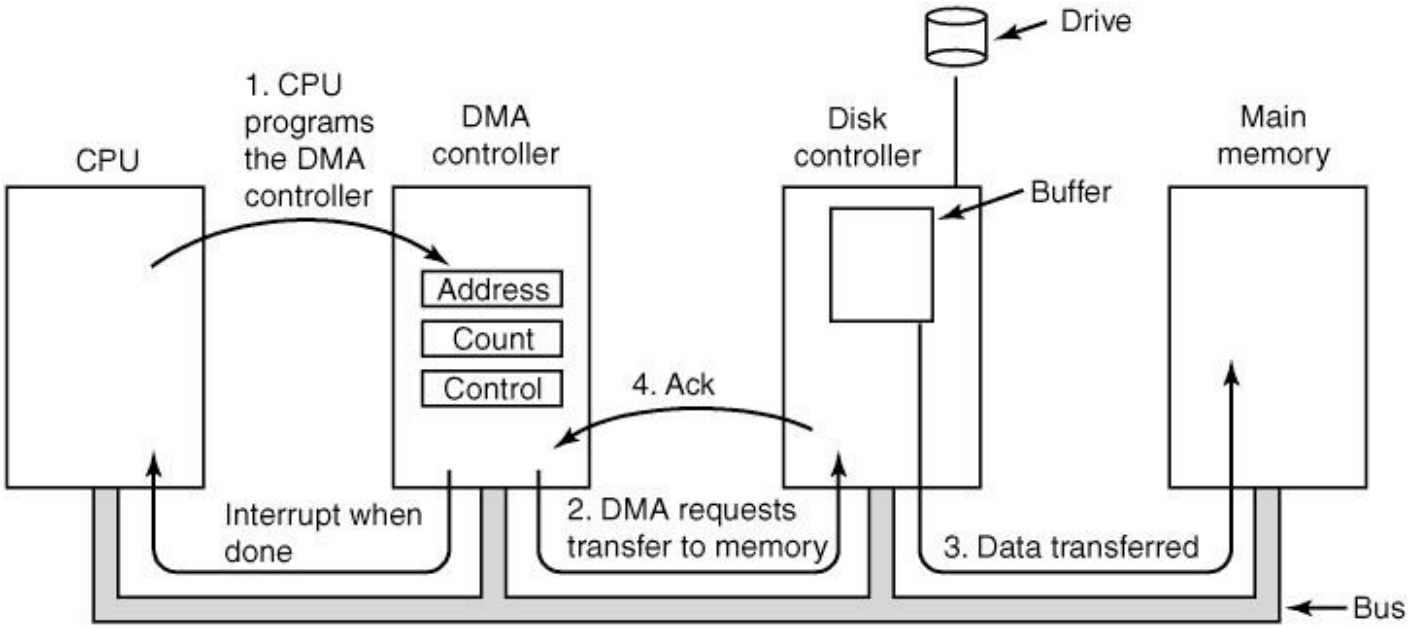
Direct Memory Access -- DMA

- **Direct Memory Access** (**Απ' Ευθείας Πρόσβαση στη Μνήμη**)
  - Ολοκλήρωση μιας εντολής I/O χωρίς την ανάμιξη της CPU.
  - Η CPU δεν χάνει χρόνο με χρονοβόρες εντολές I/O!
  - π.χ., μεταφορά ολόκληρου block δίσκου στη μνήμη, που γίνεται αποκλειστικά από τον disk controller χωρίς την ανάγκη να επέμβει η CPU να κάνει την αντιγραφή από τον buffer του controller στην μνήμη.



Direct Memory Access -- DMA

- Γιατί χρειάζονται οι **buffers** (**αποταμιευτές**) των controllers;
  - Εφόσον το system bus **μοιράζεται** ανάμεσα στη **CPU**, την **μνήμη**, και **πολλούς controllers**, μπορεί τη στιγμή που τα bits ενός μπλοκ φθάνουν σε κάποιον controller, το system bus να μην είναι διαθέσιμο (π.χ., επειδή μεταφέρει κάποια άλλη πληροφορία).
  - Ένα άλλο πρόβλημα προκύπτει από την ανικανότητα μερικών ελεγκτών για **είσοδο** και **έξοδο** **ταυτόχρονα**.

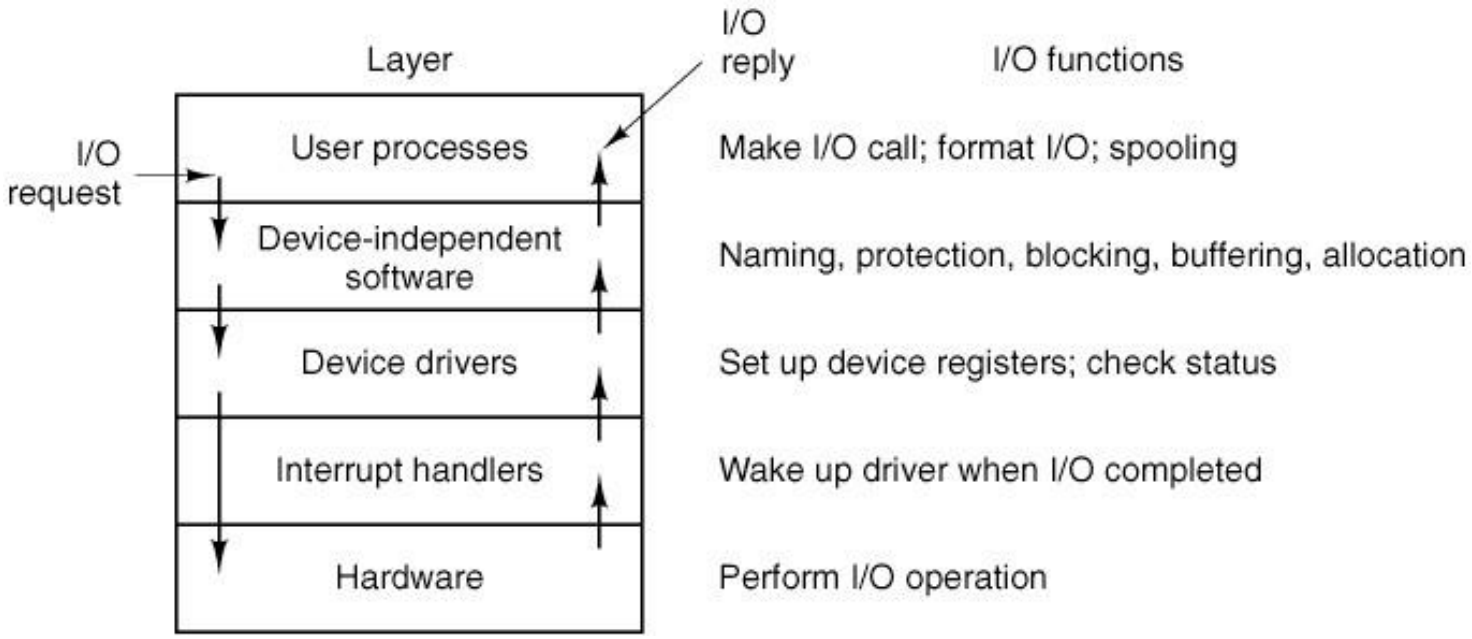


I/O Software: Βασικοί στόχοι

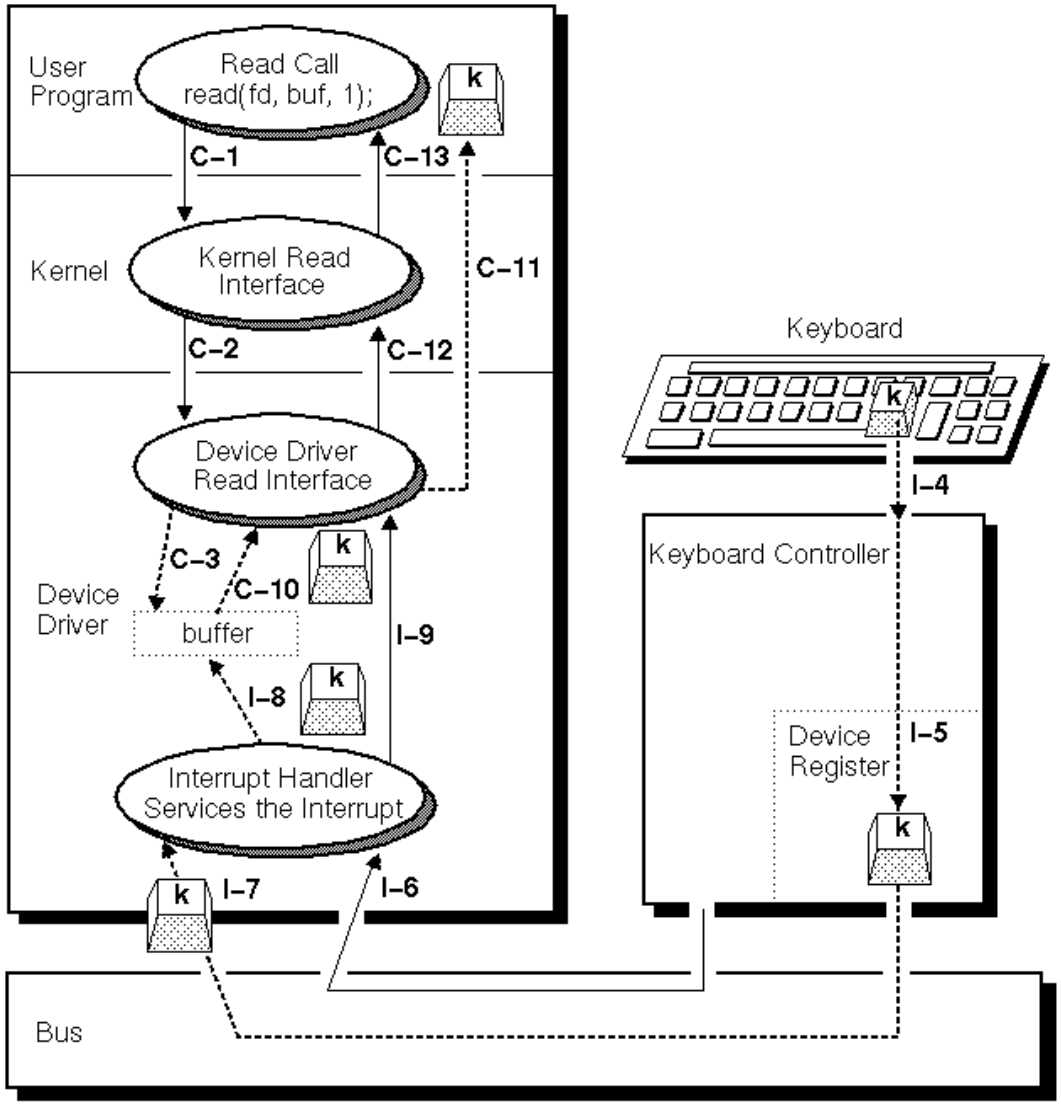
- **Device independence**
  - Όλες οι συσκευές που παρέχουν λειτουργικότητα του ίδιου είδους είναι διαχειρίσιμες με τον ίδιο ακριβώς τρόπο από τα προγράμματα.
  - Π.χ., οι ίδιες εντολές διαβάζουν από ένα hard disk, SSD, CD-ROM, floppy disk, NAS, etc.
  - Οι λεπτομέρειες υλοποίησης κάθε συσκευής είναι «κρυμμένες» από τη λειτουργικότητά τους
- **Error handling**
  - Αντιμετώπιση λαθών όσο γίνεται πιο κοντά στη συσκευή
  - Να μην απασχολούνται τα παραπάνω επίπεδα εκτός αν είναι πραγματικά απαραίτητο!
- **Uniform naming**
  - Όλες οι συσκευές ονομάζονται ομοιόμορφα.
  - Π.χ., στο Unix, όλες οι συσκευές είναι προσβάσιμες με κάποιο path name (π.χ., /dev/sda, /dev/dsp, /dev/ttyUSB0, /media/cdrom/directory/file.pdf)
- **Synchronous vs. Asynchronous I/O**
  - Οι περισσότερες λειτουργίες I/O είναι ασύγχρονες: η CPU κάνει κάτι άλλο όσο το I/O εξυπηρετείται.
  - Με τη χρήση interrupts δίνεται η ψευδαίσθηση σύγχρονης λειτουργίας.
- **Buffering**
  - Το system bus ανήκει σε πολλούς. Κάποιες φορές πρέπει να περιμένουμε να ελευθερωθεί.

I/O Software Stack

- Το I/O software είναι πολύ περίπλοκο και εξαιρετικά ποικίλο.
- Γι' αυτό, χωρίζεται σε **layers** (**επίπεδα**).



Παράδειγμα!


User Program
Read Call
read(fd, buf, 1);
k
C–1
C–13
Kernel
Kernel Read Interface
C–11
C–2
C–12
Device Driver Read Interface
C–3
C–10
k
Device Driver
buffer
I–9
k
I–8
Interrupt Handler Services the Interrupt
k
I–7
I–6
Keyboard
k
I–4
Keyboard Controller
Device Register
I–5
k
Bus

Interrupts (Διακοπές)

- Βρίσκονται στο κατώτερο επίπεδο του I/O software ώστε τα ανώτερα επίπεδα να μην απασχολούνται με όλες τις λεπτομέρειες.

- Βασική λειτουργία:
  - Όποιο process επιχειρεί I/O (για το οποίο χρειάζεται επικοινωνία με τον controller), γίνεται **blocked**.
  - Όταν ο controller τελειώσει, θα προκαλέσει interrupt και ο αντίστοιχος interrupt handler θα ξεμπλοκάρει το process (από blocked θα γίνει **waiting** (γνωστό και ως **ready**), απ΄ όπου εξετάζεται από τον CPU scheduler για να γίνει **running** όταν έρθει η σειρά του).

Device Drivers (Οδηγοί Συσκευών)

- Περιέχει τον κώδικα που είναι **device-dependent**.
- Επικοινωνεί με τον controller (μέσω των CSR του controller σ' ένα memory-mapped I/O system, όπως είδαμε πριν).
- Ο οδηγός συνήθως έχει ένα **work queue** (**ουρά εργασίας**) όπου εναποτίθενται αιτήσεις για I/O.
- Η σειρά με την οποία οι διάφορες αιτήσεις αποθηκεύονται στην ουρά ενός οδηγού είναι πολύ σημαντική για την απόδοση του συστήματος γιατί καθορίζει, π.χ., τη σειρά με την οποία τα ζητούμενα μπλοκ θ' ανακτηθούν (➔ εδώ έχουμε ένα άλλο είδος scheduling - disk scheduling).

Device Drivers (Οδηγοί Συσκευών)

- Ο driver επίσης κρατάει αρκετές πληροφορίες για τη συσκευή που διαχειρίζεται.
  - Π.χ., για έναν δίσκο πρέπει να ξέρει τη **γεωμετρία** του. Πόσοι δίσκοι (disk platters), πόσοι κύλινδροι, πόσες τροχιές (tracks) σε κάθε κύλινδρο, πόσοι τομείς (sectors) σε κάθε τροχιά, κ.λπ.
  - Επίσης πρέπει να ξέρει **σε ποιον κύλινδρο βρίσκεται τώρα η κεφαλή**
    - π.χ., για να εκτιμήσει αν χρειάζεται να ζητήσει από τον controller να κάνει seek
- Αφ' ης στιγμής εκδώσει την εντολή στον controller, **μπλοκάρει**.
- Θα τον **ξεμπλοκάρει** ο **interrupt handler** που θα προκαλέσει ο ελεγκτής όταν διεκπεραιώσει την εντολή I/O.

Device Drivers (Οδηγοί Συσκευών)

- Ευθύς αμέσως θα εξετάσει τον κατάλληλο **CSR register** για τυχόν **λάθη**.
  - Αν έχει υπάρξει λάθος, θα προσπαθήσει να το αντιμετωπίσει.
  - Αλλιώς, θα επικοινωνήσει με το αμέσως ανώτερο επίπεδο (δηλ. το device independent software) συνήθως για να του **"περάσει" πληροφορία**, όπως π.χ. ένα disk block που ανακτήθηκε, μήνυμα λάθους, κ.λπ.

- Κατόπιν, θα ελέγξει το work queue
  - Αν η ουρά εργασίας δεν είναι άδεια, θα αφαιρέσει την **επόμενη αίτηση** και θα στείλει την αίτηση στον ελεγκτή.
  - Αν είναι άδεια, θα περιμένει την επόμενη αίτηση (π.χ., θα μπλοκάρει αν ο driver είναι process) ή αλλιώς θα "επιστρέψει" (δηλ. καλεί return).

Device-Independent I/O Software

- Οι κύριες λειτουργίες σ' αυτό το επίπεδο είναι:
  - ονομασία (naming),
  - προστασία,
  - buffering,
  - ανάθεση blocks,
  - διαιτησία πρόσβασης σε "αφοσιωμένες" συσκευές
  - αναφορά λαθών

- Όπως είπαμε είναι καλό να υπάρχει uniform naming.
  - Στο UNIX κάθε device έχει ένα file name (στο οποίο αντιστοιχεί ένα ειδικό inode) - device special file.

Device-Independent I/O Software

- Το ειδικό inode περιέχει δύο αριθμούς: **major device number**, **minor device number**. Ο major device number ταυτοποιεί τον driver (π.χ. disk driver, tape driver, ...). Ο minor device number ταυτοποιεί την συσκευή που εμπλέκεται στο I/O (π.χ. ποιος δίσκος).

- Φυσικά, αφού πολλές συσκευές χρησιμοποιούνται από πολλούς χρήστες, οι συσκευές χρειάζονται **προστασία** π.χ. ο κάθε χρήστης δεν μπορεί να έχει απ' ευθείας πρόσβαση σε όποιον δίσκο θέλει (γιατί π.χ. δεν πρέπει να μπορεί να προσπελάσει ξένη πληροφορία).

Device-Independent I/O Software

- Για block devices, είπαμε ότι η **μονάδα προσπέλασης πληροφορίας** είναι το block -- το hardware δεν επιτρέπει τίποτα άλλο.

- Συνήθως το I/O software σύστημα χρειάζεται **buffers** στην Κ.Μ.
  - Χρησιμοποιώντας buffers το Λ.Σ. επιτρέπει σε χρήστες (user processes) να προσπελάσουν οποιοδήποτε τμήμα πληροφορίας έχει ήδη διαβαστεί στη μνήμη.
  - Σε **block devices**, όπως δίσκοι, αυτό το σύστημα buffering έχει ευεργετικές συνέπειες για την απόδοση του συστήματος. Γιατί, αν το ζητούμενο block είναι σ' ένα buffer τότε αποφεύγεται το disk I/O.
  - Σε **character devices**, οι buffers χρειάζονται π.χ. γιατί input από το πληκτρολόγιο μπορεί να φθάσει πριν ζητήσει να το διαβάσει η εφαρμογή.

Device-Independent I/O Software

- Ο αλγόριθμος και οι δομές δεδομένων, για να αναθέτουν **ελεύθερα (free) disk blocks**, είναι ένα άλλο μέρος του device independent I/O s/w.

- Μερικές συσκευές (π.χ. εκτυπωτές) μπορούν να χρησιμοποιηθούν μόνο από ένα process κάθε φορά. Έτσι, αιτήσεις για μη-διαθέσιμες συσκευές είτε απορρίπτονται είτε μπλοκάρουν.

- Όταν ο driver δεν μπορέσει να αντιμετωπίσει κάποιο λάθος (συνήθως, προσπαθώντας/επαναλαμβάνοντας το I/O call μερικές φορές) τότε ειδοποιεί το device independent I/O software για το λάθος. Αυτό, ανάλογα με το είδος του λάθους, είτε το αναφέρει στο User process, είτε τερματίζει το Λ.Σ. (για πιο σοβαρά λάθη).

User-level I/O Software

- Ένα μεγάλο μέρος αυτού του τμήματος περιέχει τις **library routines** (π.χ. **read()**, **write()**, **open()**, **close()**, **seek()**,...).

- Ενα άλλο μεγάλο μέρος είναι **spooling software**
  - Αυτό το λογισμικό έχει την ευθύνη να διαχειρίζεται αφοσιωμένες συσκευές σε multiprogrammed συστήματα (π.χ. εκτυπωτές: μόνο ένας χρήστης πρέπει να μπορεί να εκτυπώνει κάθε φορά).
  - Ο spooler είναι ένα ειδικό **process daemon** (δαίμονας) που διαχειρίζεται ένα spooling directory. Για να εκτυπωθεί ένα αρχείο, το process το βάζει στο spooling directory. Ο spooler είναι το μόνο process του printer. Ο spooler μόλις τελειώσει μια αίτηση, παίρνει ένα άλλο αρχείο από το spooling directory και το εκτυπώνει.

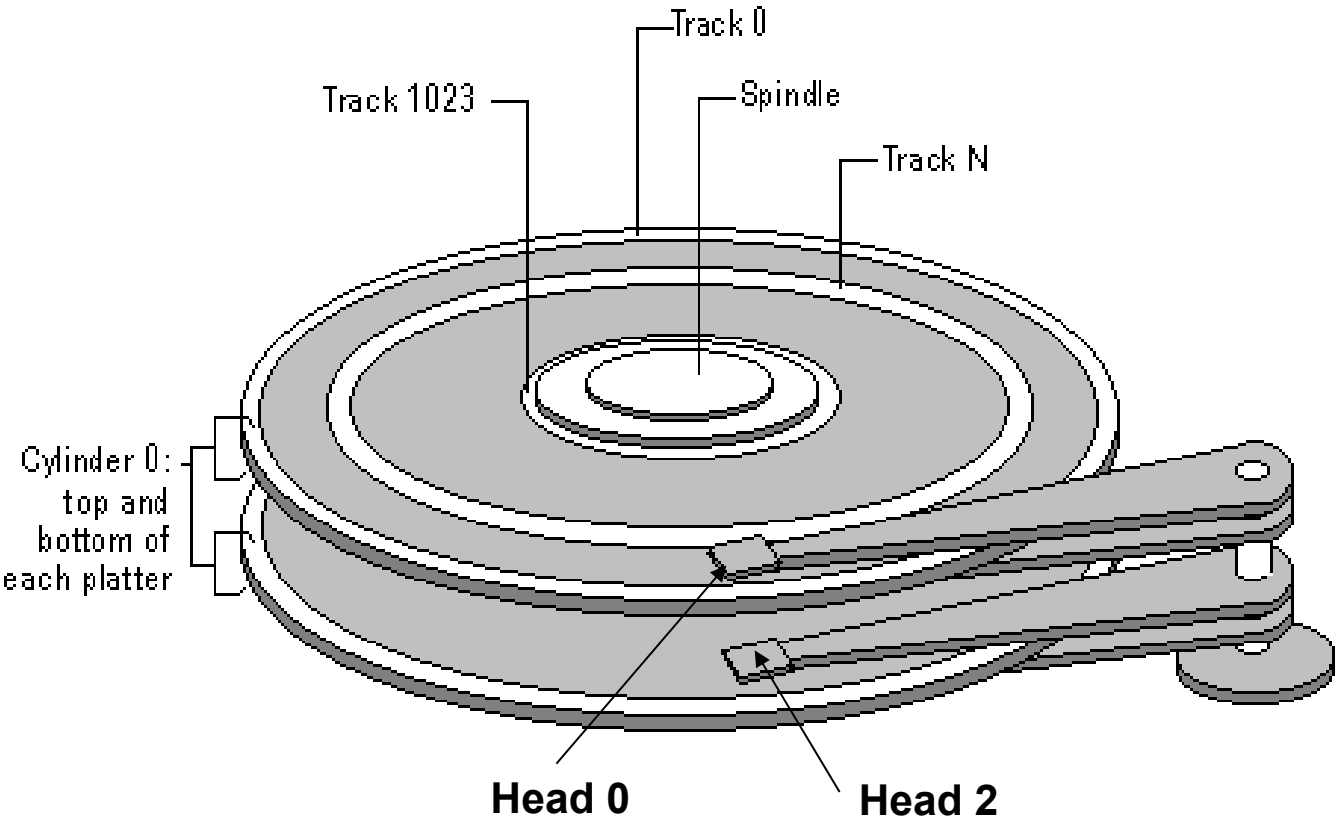
Μαγνητικοί Δίσκοι

- Από τις πιο σημαντικές συσκευές ενός σύγχρονου υπολογιστή (…ακόμα!)

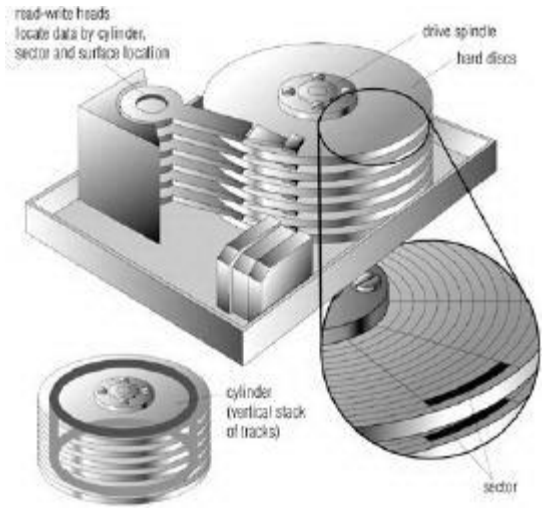
- Το πλεονέκτημά τους:
  - πολύ μεγαλύτερη χωρητικότητα από την Κ.Μ.
  - πιο φθηνή μνήμη
  - "σταθερή" μνήμη (non-volatile): όταν καταρρεύσει η μηχανή, τα περιεχόμενα δεν χάνονται

- Γεωμετρία:
  - Κύλινδροι (πχ συνήθως 6000 – 10000) περιέχουν τροχιές (tracks) (πχ συνήθως 8 – 12) που περιέχουν τομείς (sectors) (πχ συνήθως 120—300).
  - Κάθε δίσκος έχει 2 επιφάνειες. Σε κάθε επιφάνεια αντιστοιχεί μία κεφαλή.
  - Οι τομείς περιέχουν τον ίδιο αριθμό bytes (πχ 512).

Ο σκληρός δίσκος

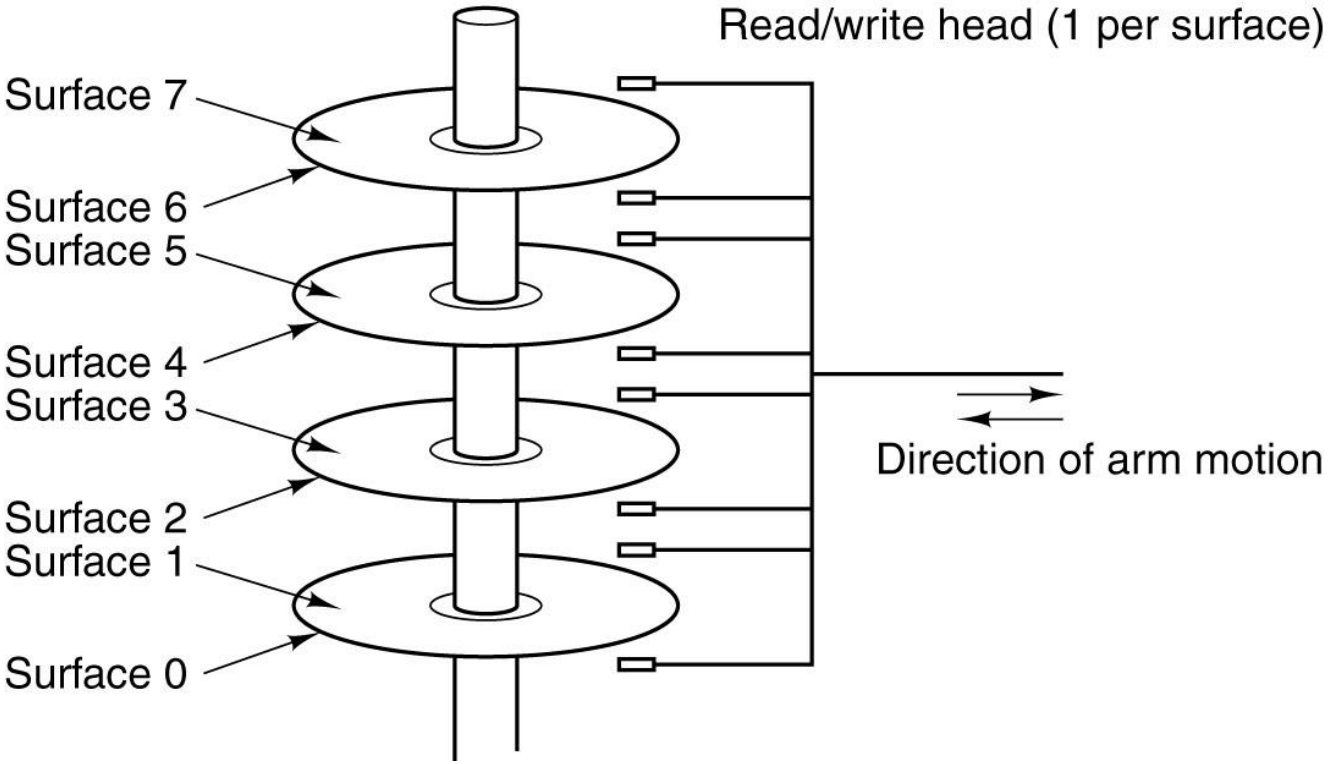


Ο σκληρός δίσκος



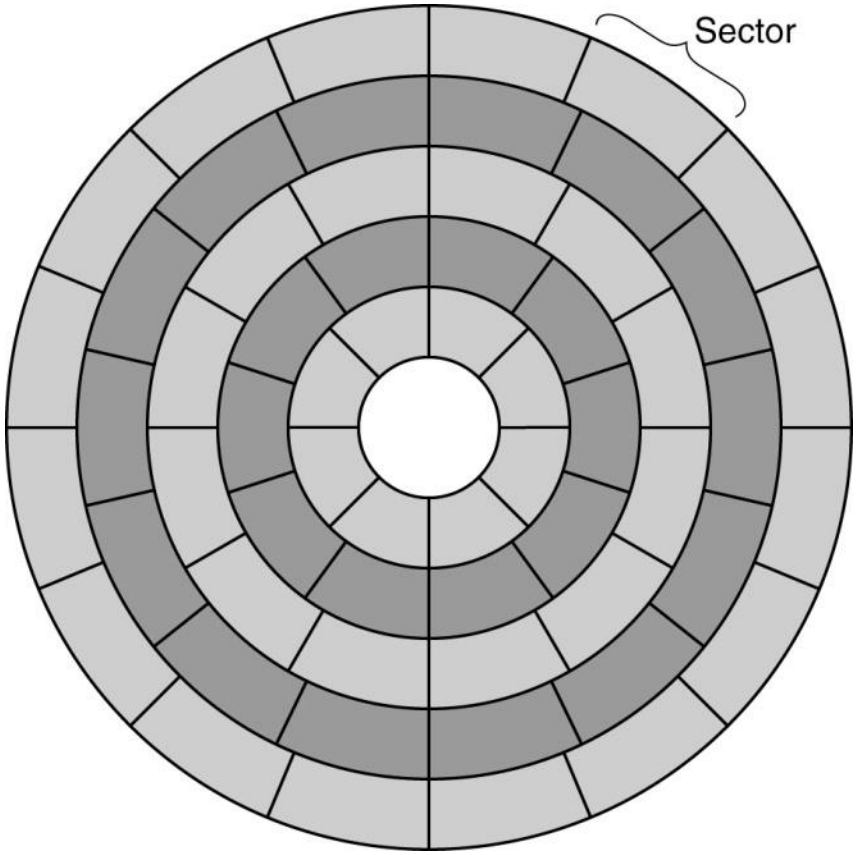
Ο σκληρός δίσκος

- Κάθε δίσκος έχει:
  - **6-12 platters** (5400-10000 περιστροφές ανά λεπτό)
  - **1 arm** για να διαβάζει/γράφει δεδομένα



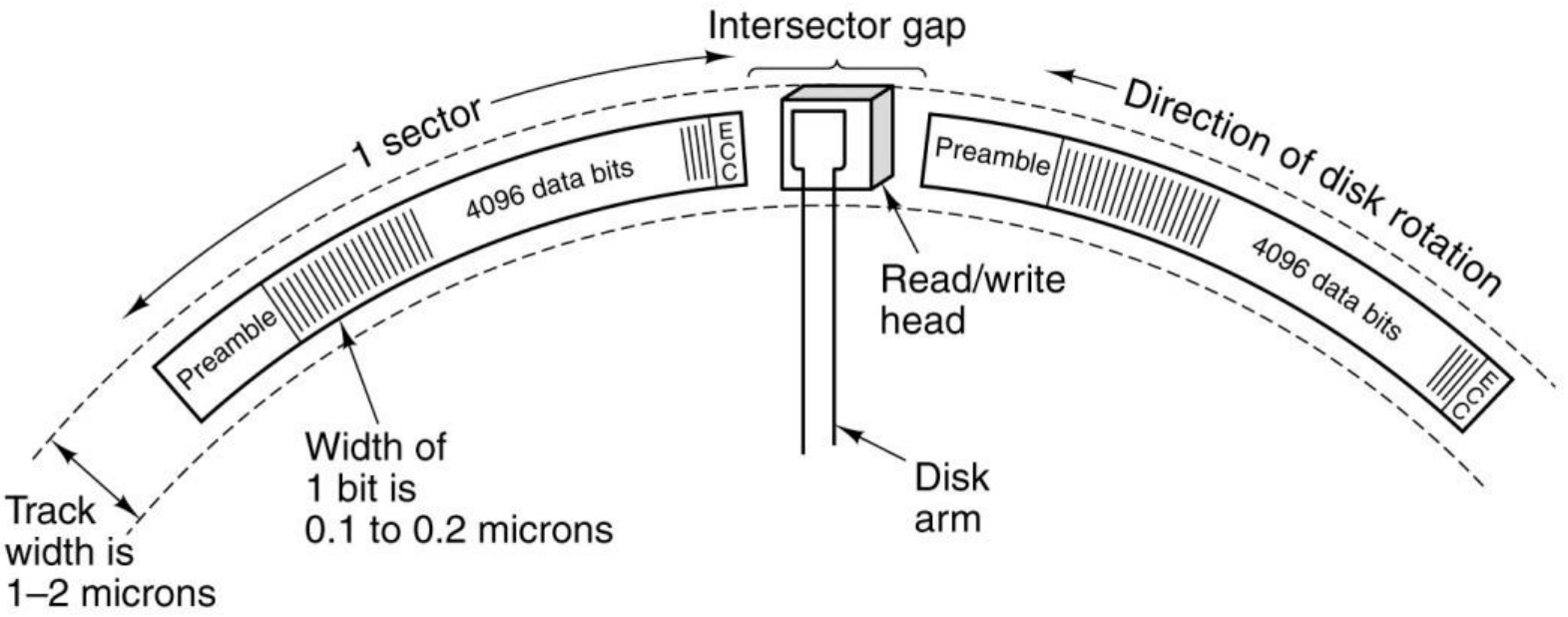
Ο σκληρός δίσκος

- Κάθε platter έχει **cylinders** (**κυλίνδρους**)
- Κάθε cylinder έχει πολλά **sectors**



Ο σκληρός δίσκος

- Κάθε sector περιέχει
  - Ένα **preamble** (για να αναγνωρίζει η κεφαλή την αρχή του)
  - **512 Bytes** of data
  - Ένα **error correction code** (π.χ., Hamming code)



Μαγνητικοί Δίσκοι

- Σχετικά πρόσφατα παρουσιάστηκαν δίσκοι, όπου τα εξωτερικά tracks έχουν πιο πολλά sectors) -- **zoned disks**.
  - ➔ μεταβαλλόμενος ρυθμός μεταφοράς δεδομένων!

- Συνήθως, όταν ένας controller ελέγχει *> 1* δίσκο, τότε υπάρχει η δυνατότητα για **επικαλυπτόμενες αναζητήσεις** (**overlapped seeks**)
  - ο controller ζητάει από ένα δίσκο ένα seek και πριν τελειώσει ζητά και 2ο seek από άλλον δίσκο).
  - Επίσης συχνά μπορεί να ζητηθεί ένα read/write από έναν δίσκο καθ' όσον άλλος δίσκος εκτελεί μια εντολή seek.
  - Δυστυχώς, όμως, παράλληλα read/write σε δύο δίσκους ενός controller, δεν είναι δυνατά.

Φυσικά, τα παραπάνω έχουν ευεργετικές συνέπειες για την απόδοση του συστήματος.

Disk Scheduling

- **Αλγόριθμοι Χρονοπρογραμματισμού Δίσκων**
  - Στόχος είναι η βελτιστοποίηση της απόδοσης
- Λογισμικό
  - ο αλγόριθμος με βάση τον οποίο **ο οδηγός δίσκου επιλέγει την επόμενη αίτηση** για να εξυπηρετήσει ο δίσκος – θυμηθείτε την ουρά εργασίας του οδηγού δίσκων.
- Υλικό
  - ο ελεγκτής συνήθως κάνει **βελτιστοποιήσεις**
- Τα **performance metrics** (μετρικές απόδοσης) περιλαμβάνουν:
  - **Response time** (Χρόνος απόκρισης)
  - **Throughput** (Ρυθμοαπόδοση)
  - **Fairness** (Δικαιοσύνη, π.χ., αποφυγή λιμοκτονίας)

33

Disk Scheduling

- Βασικά συστατικά κόστους προσπέλασης στον δίσκο:
  - **Χρόνος αναζήτησης -- seek time**: κόστος μετακίνησης του βραχίονα (με τις κεφαλές) στον σωστό κύλινδρο.
    - Εξαρτάται από την απόσταση που διανύεται ➔ συμφέρουν "μικρά" seeks.
    - Συνήθως 5-20 msec
  - **Χρόνος περιστροφής -- rotational delay**: ο χρόνος που απαιτείται (μετά το seek) ώστε το επιθυμητό sector να έρθει κάτω από την κεφαλή.
    - Με 7200rpm, μια πλήρης περιστροφή παίρνει 8.3 msec
  - **Χρόνος μεταφοράς -- transfer time**: ο χρόνος που απαιτείται για την ανάκτηση/μεταφορά δεδομένων
    - Συνήθως < 100 μsec.

Disk Scheduling

- Πως μπορούμε να μειώσουμε το χρόνο disk I/O;
  - Πιο γρήγορη περιστροφή
  - Τοποθέτηση σχετικών δεδομένων σε γειτονικές θέσεις
  - Μείωση seek time με έξυπνο scheduling όταν εκκρεμούν παράλληλες αιτήσεις

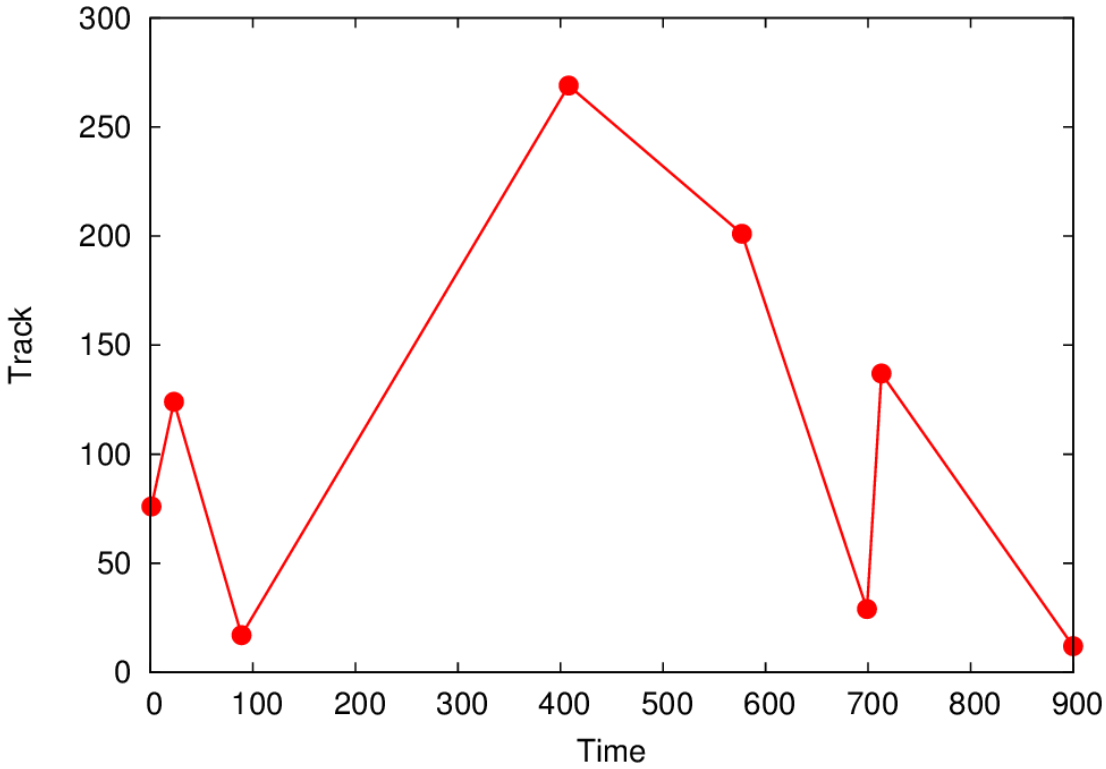
- Οι περισσότεροι γνωστοί αλγόριθμοι υποθέτουν ότι το seek time ευθύνεται για το μεγαλύτερο κόστος

➔ αυτό είναι το κόστος που οι αλγόριθμοι προσπαθούν να ελαχιστοποιήσουν.

- Για να επιτύχουμε αυτόν το στόχο πρέπει να αξιοποιήσουμε τις εξής γνώσεις των driver/controller:
  - τωρινή θέση (δηλ. αριθμός κυλίνδρου) των κεφαλών
  - επιθυμητός κύλινδρος για κάθε αίτηση στην ουρά

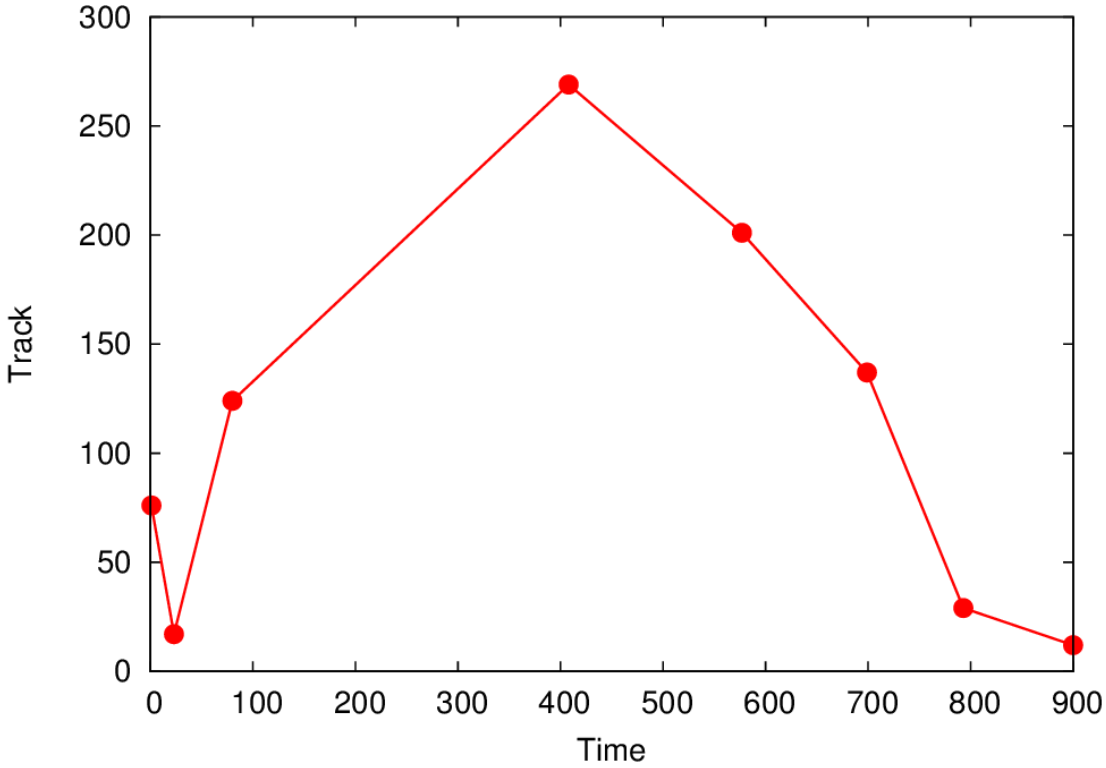
First-In, First-Out (FIFO)

- Εξυπηρέτηση αιτήσεων με τη σειρά άφιξης
  - Απλό
  - Αργό (πολλές κινήσεις της κεφαλής)!



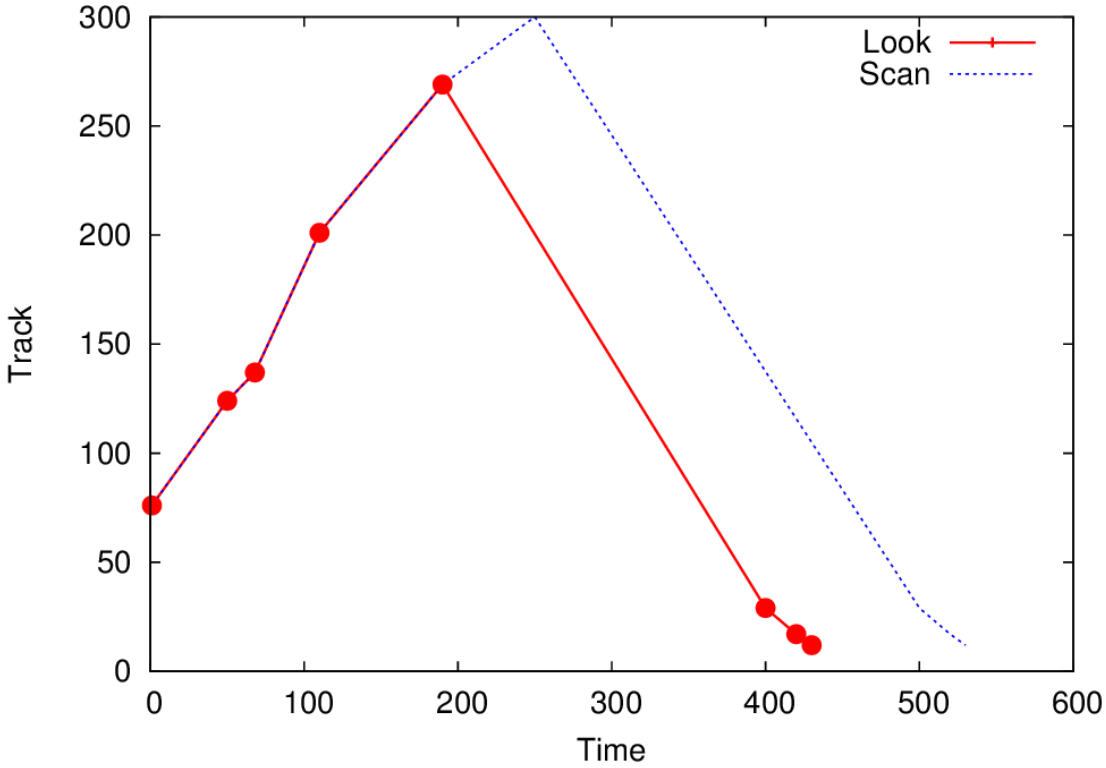
Shortest Seek Time First (SSTF)

- Εξυπηρετούμε πάντα την πιο κοντινή αίτηση
  - Πιο μικρές κινήσεις → **πιο γρήγορο**
  - **Ελαχιστοποιεί το seek time**
  - Μπορεί όμως να οδηγήσει σε **starvation**, κυρίως εις βάρος των άκρων



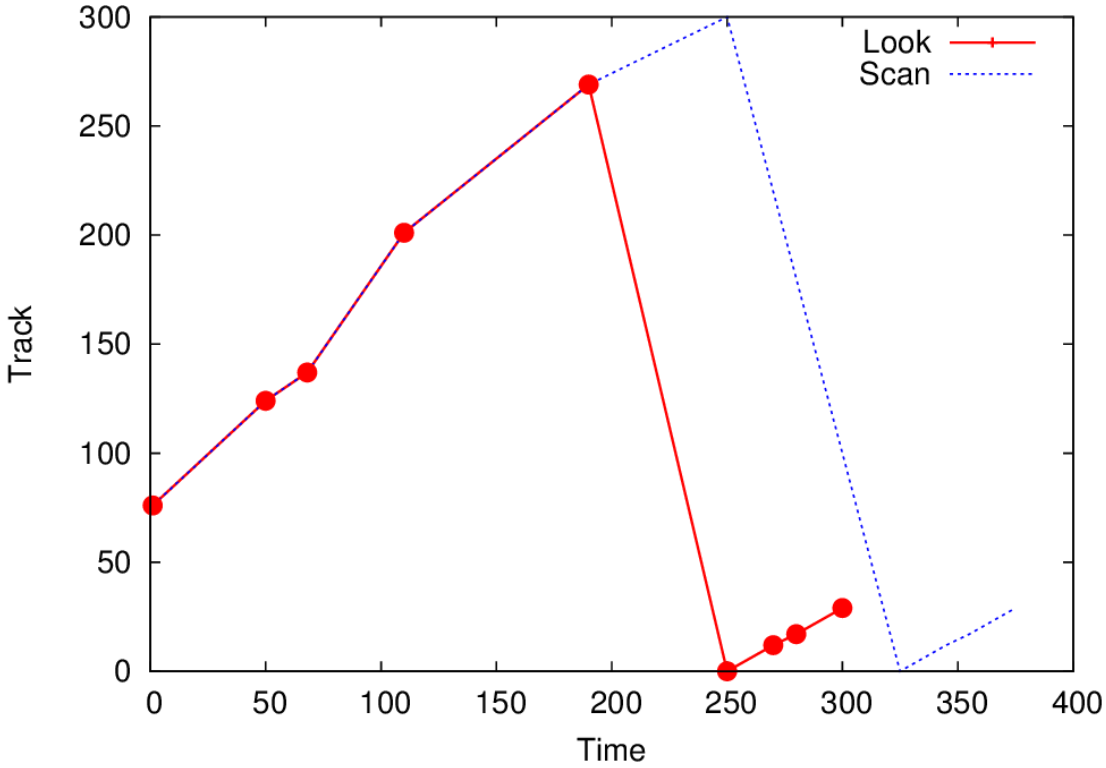
Scan και Look (ή Elevator)

- Σαν τον SSTF, αλλά υπάρχει η έννοια της **κατεύθυνσης**: όσο υπάρχουν αιτήσεις «προς την ίδια κατεύθυνση», τις εξυπηρετούμε.
  - Λίγο **χειρότερος** από τον SSTF όσον αφορά το seek time.
  - **Πιο δίκαιος** από τον SSTF
    - …αλλά όχι απόλυτα. Γιατί;;;;



Circular SCAN (ή C-SCAN)

- Σαν τον SCAN, αλλά το «σκούπισμα» γίνεται πάντα προς την ίδια κατεύθυνση
  - Ακόμη πιο δίκαιος από τον SCAN



Disk Scheduling

- Βελτίωση rotation time
  - Περαιτέρω πιθανότητες βελτίωσης υπάρχουν όταν εκκρεμούν αιτήσεις για διαφορετικά sectors του ίδιου κυλίνδρου.
  - Όταν ο controller εξάγει την πληροφορία για το ποια τομή περνά κάτω από την κεφαλή τότε ο οδηγός εξυπηρετεί πρώτα την αίτηση για την τομή που θα περάσει κάτω από την κεφαλή πρώτη.

Disk Scheduling

- Όταν ο ίδιος driver διαχειρίζεται πολλούς disk controllers:
  - Όσο γίνεται κάποιο transfer, ο driver εκδίδει **εντολές seek στα άλλα drives** ώστε να γίνουν όσο το δυνατόν περισσότερες λειτουργίες παράλληλα.
  - Όταν μια εντολή εγγραφής ή ανάκτησης τελειώσει, ο driver μπορεί να ελέγξει αν για κάποιο drive ισχύει ότι οι κεφαλές του είναι τοποθετημένες σε κύλινδρο για τον οποίο υπάρχει αίτηση στο work queue του οδηγού και έτσι δρομολογείται αυτή η αίτηση αφού δεν χρειάζεται να γίνει seek.
  - Κατά πάσα πιθανότητα το seek έγινε παράλληλα με την προηγούμενη ανάκτηση/εγγραφή

Disk Scheduling

- Πρέπει να σημειωθεί ότι οι πιο πολυ-υλοποιημένοι αλγόριθμοι βασίζονται σε παραλλαγές του SCAN. Δηλαδή γίνεται προσπάθεια ελαχιστοποίησης του seek time.

- Η τεχνολογία δίσκων ήδη έχει αλλάξει (SSD) και το seek time έχει παύσει να είναι το κυρίαρχο κόστος ➔ ανάγκη για καινούργιους αλγόριθμους για disk scheduling.

- Μερικοί αλγόριθμοι προσπαθούν να ελαχιστοποιήσουν το άθροισμα του χρόνου αναζήτησης και περιστροφής

- Βελτιστοποιήσεις ελεγκτή:
  - **Sector prefetching**: μετά την αναζήτηση κυλίνδρου και την καθυστέρηση περιστροφής, δεν ανακτάται μόνο το ζητούμενο sector (τομέας) αλλά και μερικά επόμενα ή και όλα τα sectors του κυλίνδρου.
  - Τα επιπλέον sectors τοποθετούνται στην **κρυφή μνήμη του ελεγκτή** και ανακτώνται από εκεί αν ζητηθούν (➔ αποφεύγεται το κόστος πρόσβασης στη μαγνητική επιφάνεια του δίσκου).