Προγραμματισμός II (ECE116)

#12

σύστημα αρχείων (file system)

Σπύρος Λάλης
lalis@uth.gr

Μόνιμη αποθήκευση δεδομένων

- Οι μεταβλητές και δομές δεδομένων ενός προγράμματος υπάρχουν στην μνήμη του Η/Υ
- **Χάνονται** όταν τερματιστεί το πρόγραμμα ή σβήσει ο Η/Υ (πιθανώς λόγω βλάβης)

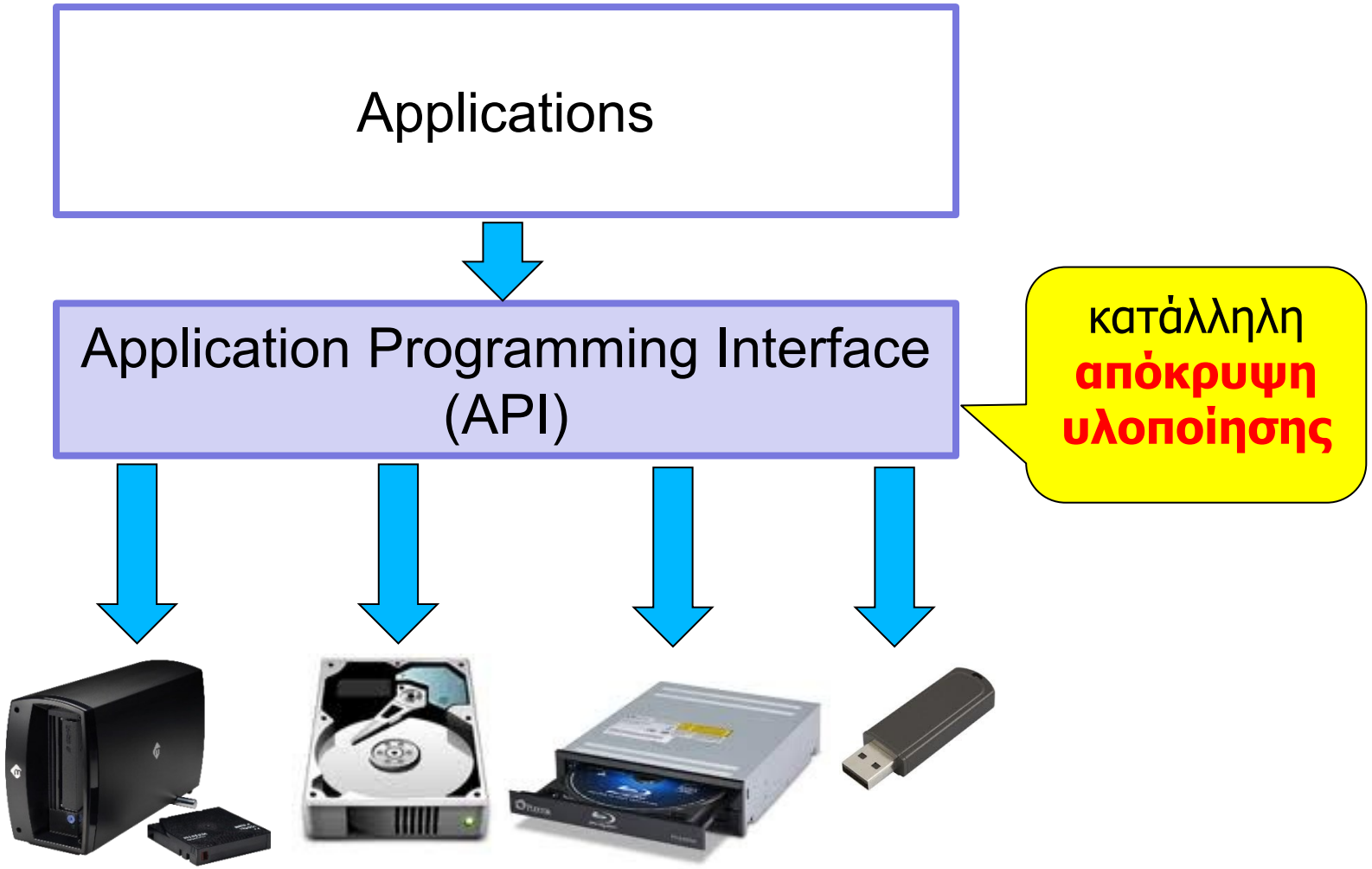
- Χρειαζόμαστε **μόνιμα** αποθηκευτικά μέσα που λειτουργούν **χωρίς** παροχή ρεύματος
  - μαγνητικά, οπτικά, solid-state, …

- Σημείωση: και ο ίδιος ο κώδικας (sources, binaries) είναι δεδομένα που δεν θέλουμε να χάνονται κάθε φορά που σβήνουμε τον υπολογιστή

Πρόσβαση μέσων αποθήκευσης

- Η πρόσβαση στα μόνιμα αποθηκευτικά μέσα είναι μια αρκετά περίπλοκη υπόθεση
  - καμία σχέση με το απλό μοντέλο της κυρίως μνήμης
  - κάθε τεχνολογία έχει τις ιδιαιτερότητες της
  - κάθε συσκευή μπορεί να απαιτεί διαφορετική διαχείριση

- Χρειάζεται ένα **γενικό μοντέλο πρόσβασης ανεξάρτητου** της τεχνολογίας που χρησιμοποιείται για την υλοποίηση μιας αποθηκευτικής συσκευής
- Επίσης χρειάζεται **ελεγχόμενη πρόσβαση** στις συσκευές αποθήκευσης, ώστε να παρέχεται **προστασία/ασφάλεια** των δεδομένων


Applications
Application Programming Interface (API)
κατάλληλη απόκρυψη υλοποίησης

Αρχεία (files)

- Ξεχωριστές **οντότητες αποθήκευσης** δεδομένων

- Τα αρχεία είναι **μόνιμα**
  - εξακολουθούν να υφίστανται ακόμα και μετά τον τερματισμό των προγραμμάτων που τα δημιούργησαν/επεξεργάστηκαν
  - συχνά, επιζούν πολύ παραπάνω και από τα προγράμματα και τους υπολογιστές μέσω των οποίων δημιουργήθηκαν

- Τα αρχεία είναι **επώνυμα**
  - κάθε αρχείο έχει «μοναδικό» **όνομα**
  - για να μπορεί να **εντοπιστεί** εύκολα
  - από τον χρήστη (και από προγράμματα)

Σύστημα αρχείων (file system)

- Το λογισμικό που είναι υπεύθυνο (α) για την **υλοποίηση** της λειτουργικότητας των αρχείων, καθώς και (β) για την **διαχείριση** των αρχείων
  - συνήθως κομμάτι του λειτουργικού συστήματος

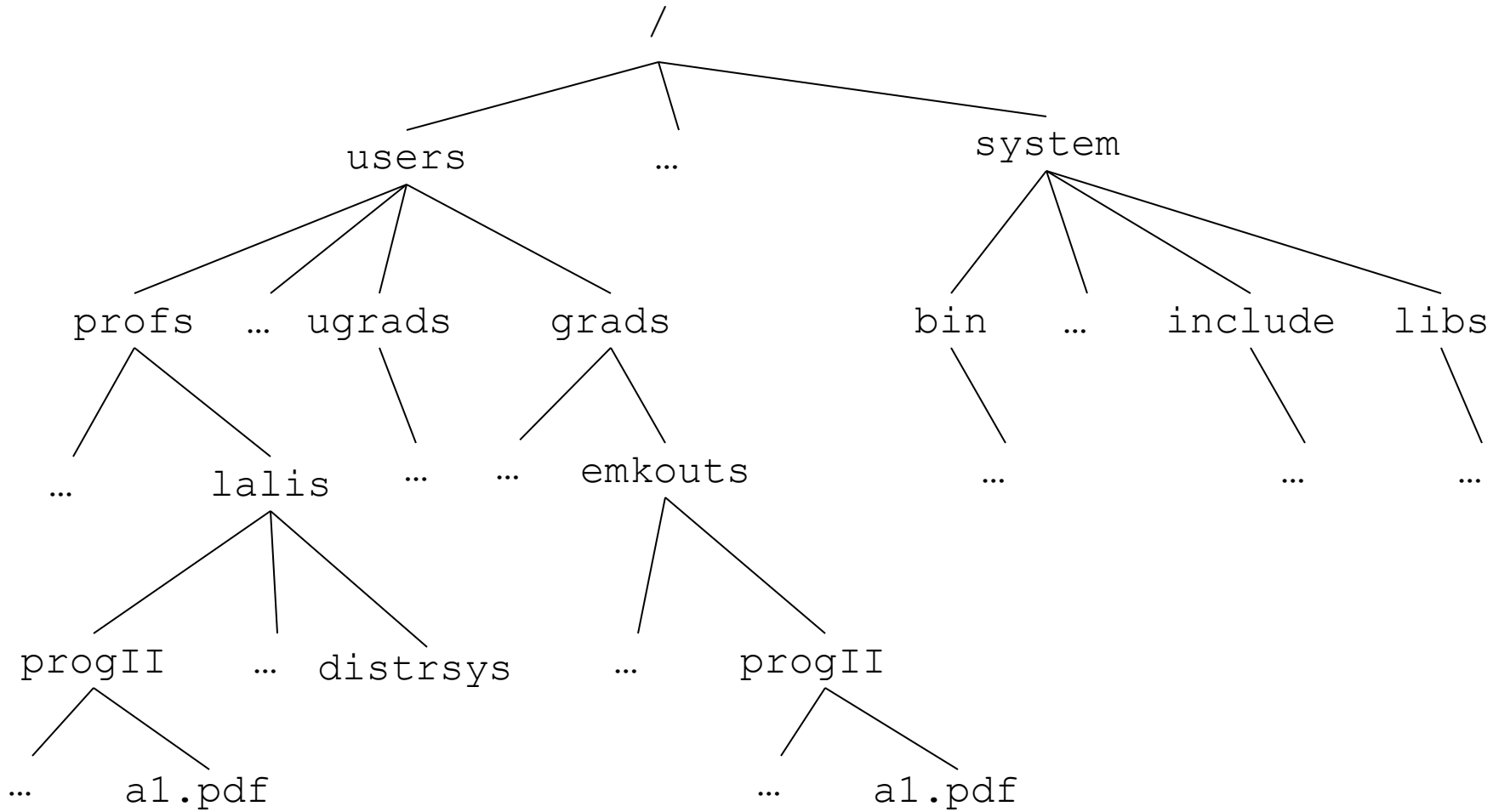
- Παρέχει στις εφαρμογές μια απλή **αφαιρετική** διεπαφή προγραμματισμού (file system API)
  - κρύβει όλη την πολυπλοκότητα υλοποίησης

Βασικές πτυχές λειτουργικότητας:

1. **Διαχείριση χώρου ονομάτων**
2. **Προσπέλαση δεδομένων** (δίσκος <-> εφαρμογή)

Χώρος ονομάτων (name space)

- Τα ονόματα των αρχείων δίνονται στο πλαίσιο του **χώρου ονομάτων** του συστήματος αρχείων
- Ο χώρος είναι δομημένος **ιεραρχικά**
  - διευκολύνει την ανάθεση ονομάτων χωρίς «συγκρούσεις»
  - διευκολύνει την διαχείριση πολλών αρχείων
  - διευκολύνει τον έλεγχο πρόσβασης
- Κάθε επίπεδο ονομάζεται **κατάλογος** (directory)
- Η δομή αναπαρίσταται ως «ανεστραμμένο» δέντρο
- Η ρίζα (root) είναι στο ψηλότερο επίπεδο και αποτελεί την κορυφή της ιεραρχίας καταλόγων
- Τα αρχεία μπορεί να δημιουργηθούν/τοποθετηθούν σε οποιοδήποτε επίπεδο του χώρου ονομάτων

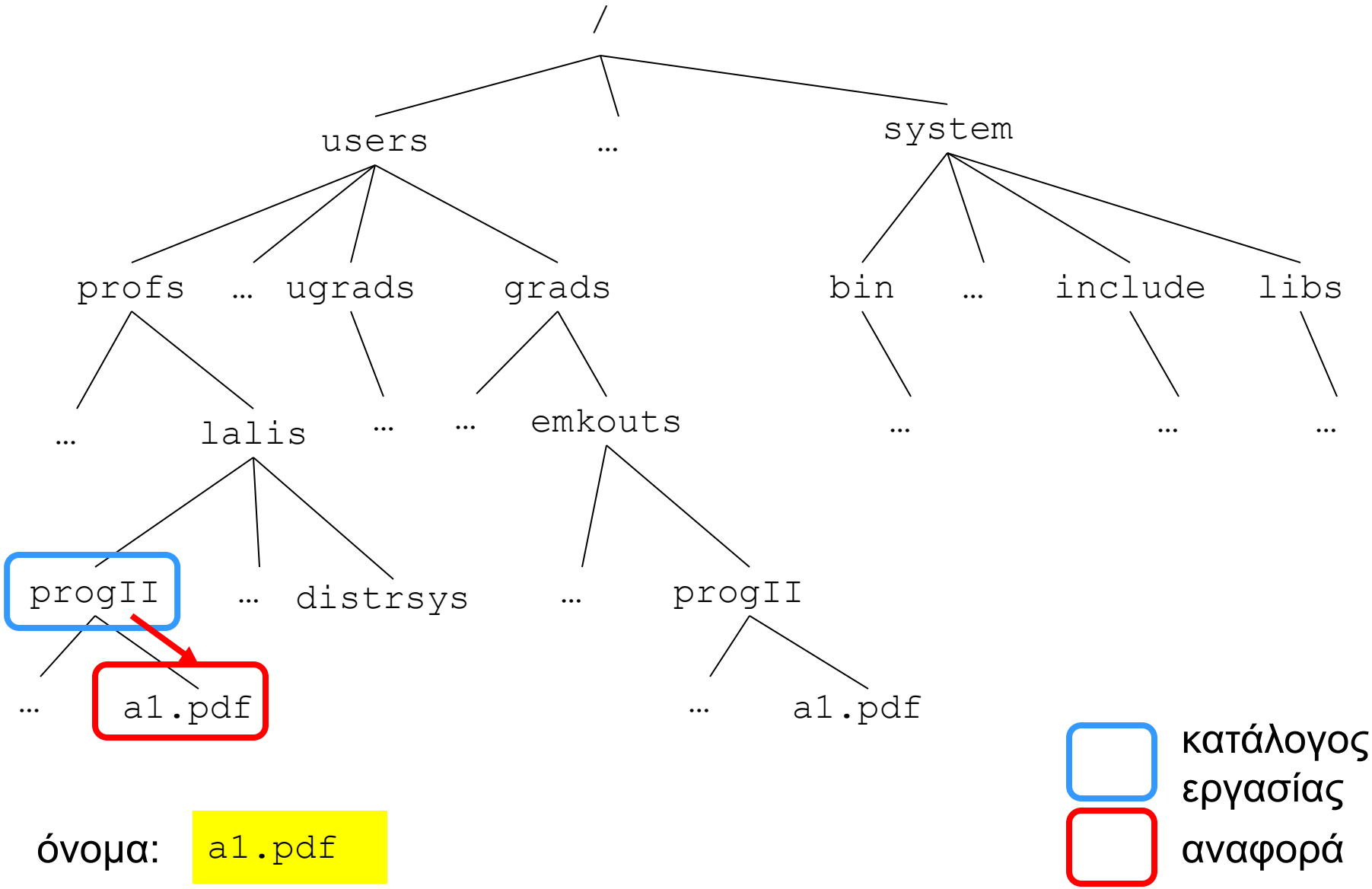

/
users
…
system
profs
…
ugrads
grads
bin
…
include
libs
…
lalis
…
…
emkouts
…
…
…
progII
…
distrsys
…
progII
…
a1.pdf
…
a1.pdf

Προσδιορισμός ονομάτων

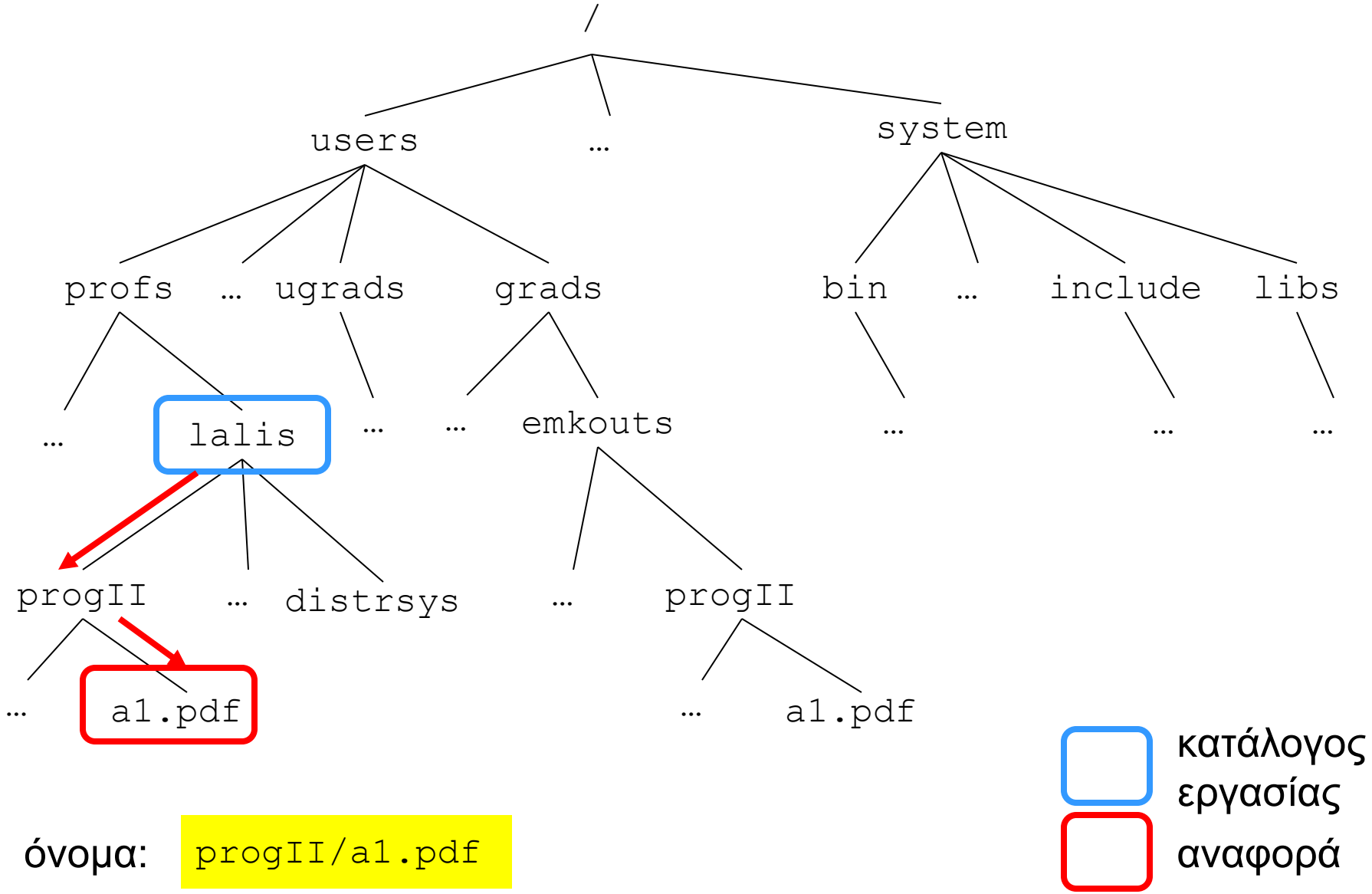
- **Απόλυτος** προσδιορισμός
  - όνομα με βάση την **κορυφή της ιεραρχίας**
- **Σχετικός** προσδιορισμός
  - όνομα με βάση τον **τρέχοντα κατάλογο εργασίας**

Συμβάσεις «πλοήγησης» μέσα στην ιεραρχία

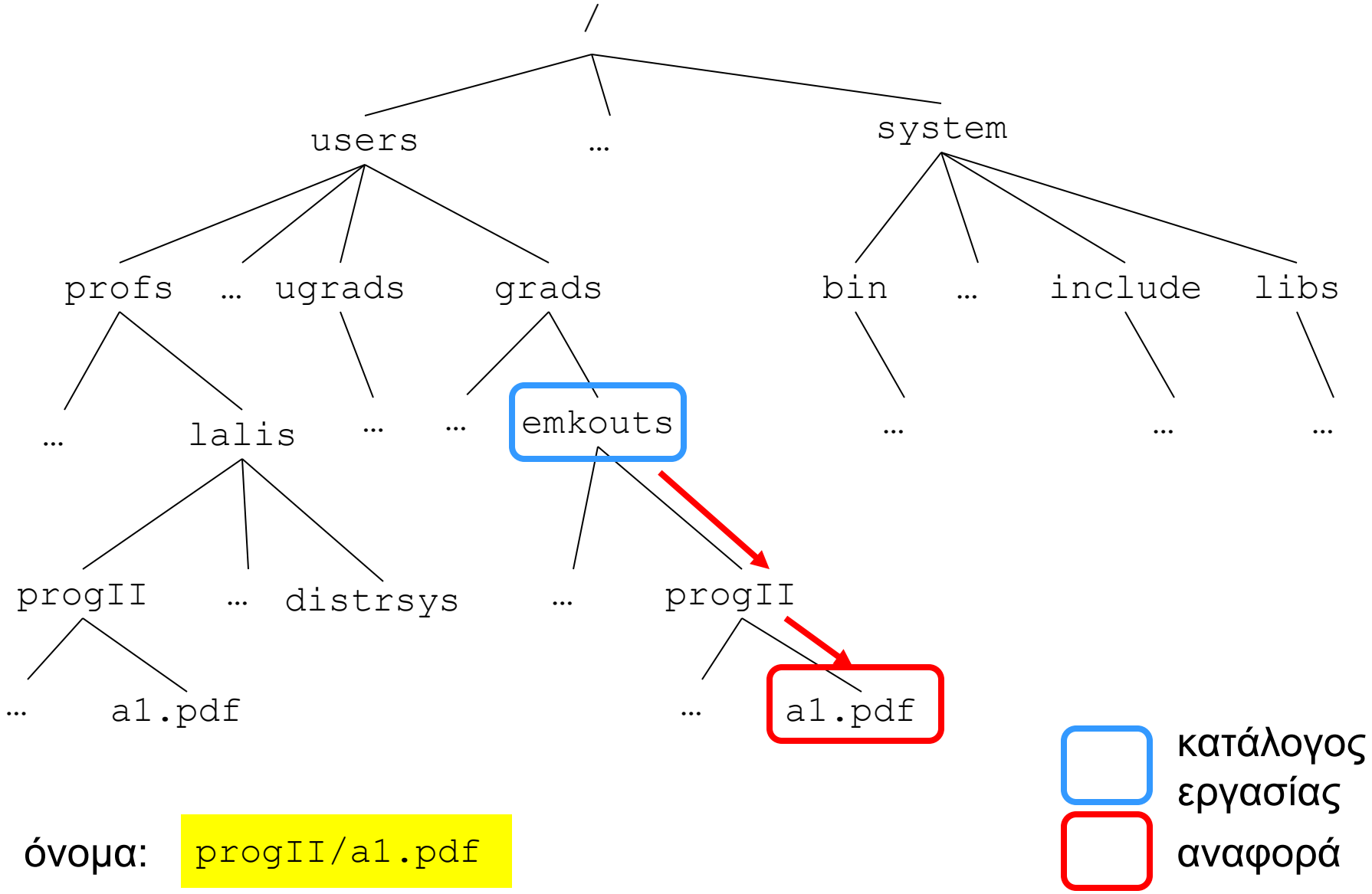
- `"/"` για την ρίζα / κορυφή της ιεραρχίας, και ως διαχωριστικό ανάμεσα στα ονόματα καταλόγων/αρχείων ενός μονοπατιού
- `"."` για τον τρέχοντα κατάλογο εργασίας
- `".."` για τον κατάλογο που βρίσκεται ένα επίπεδο ψηλότερα από τον τρέχοντα κατάλογο εργασίας


/
users
…
system
profs
…
ugrads
grads
bin
…
include
libs
…
lalis
…
…
emkouts
…
…
…
progII
…
distrsys
…
progII
…
a1.pdf
…
a1.pdf
κατάλογος εργασίας
αναφορά

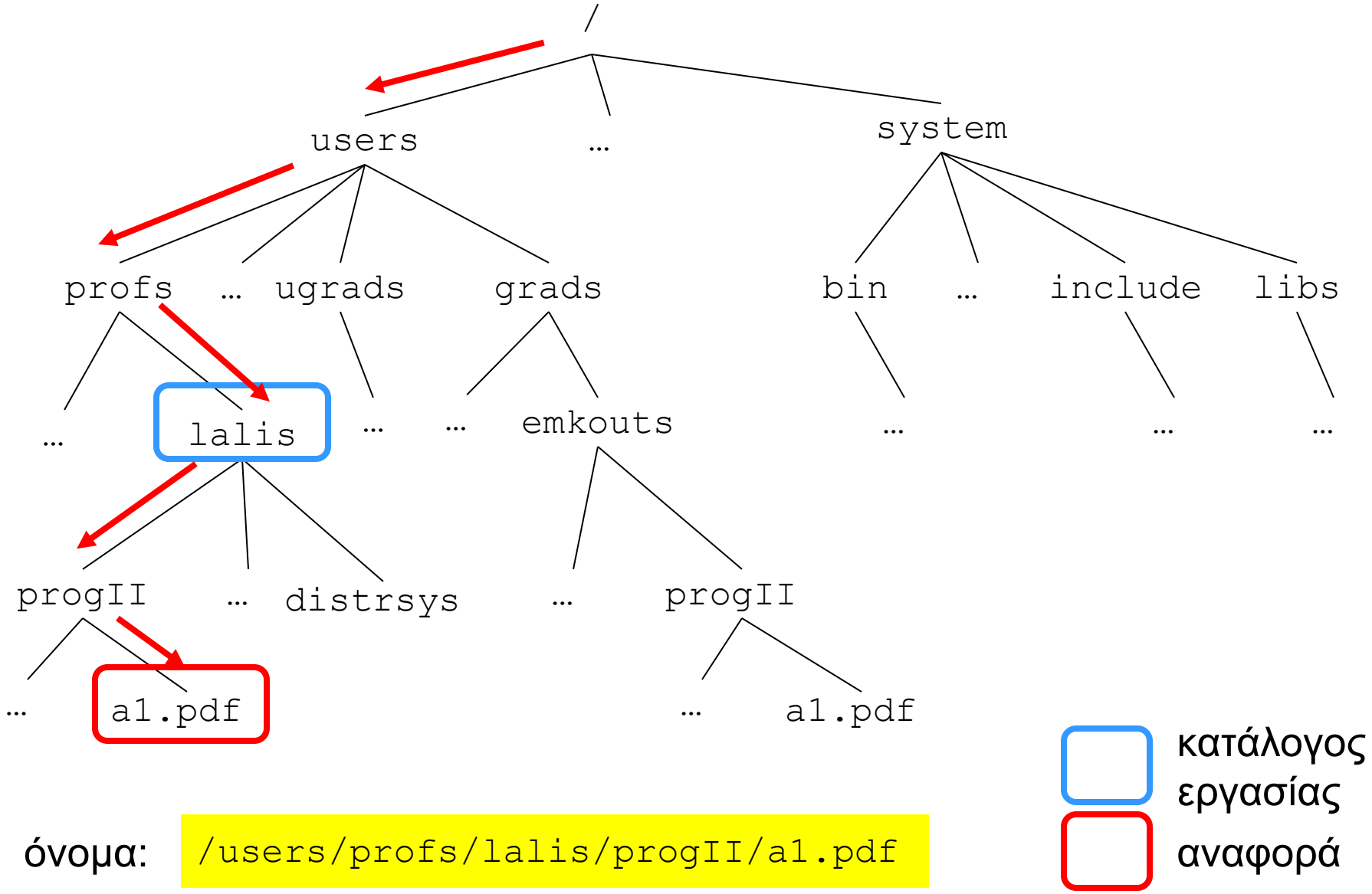
όνομα: `a1.pdf`



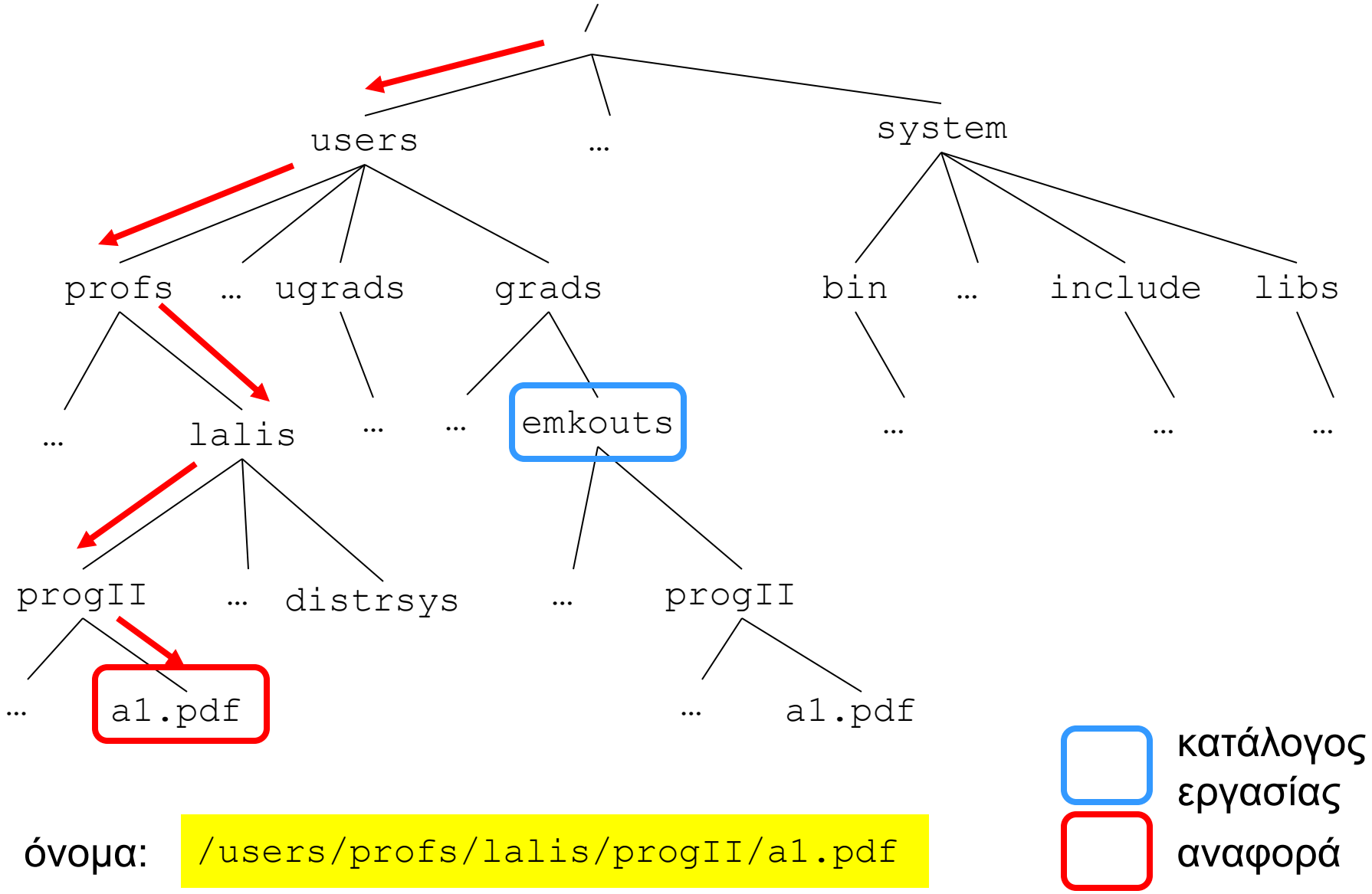
όνομα: `progII/a1.pdf`



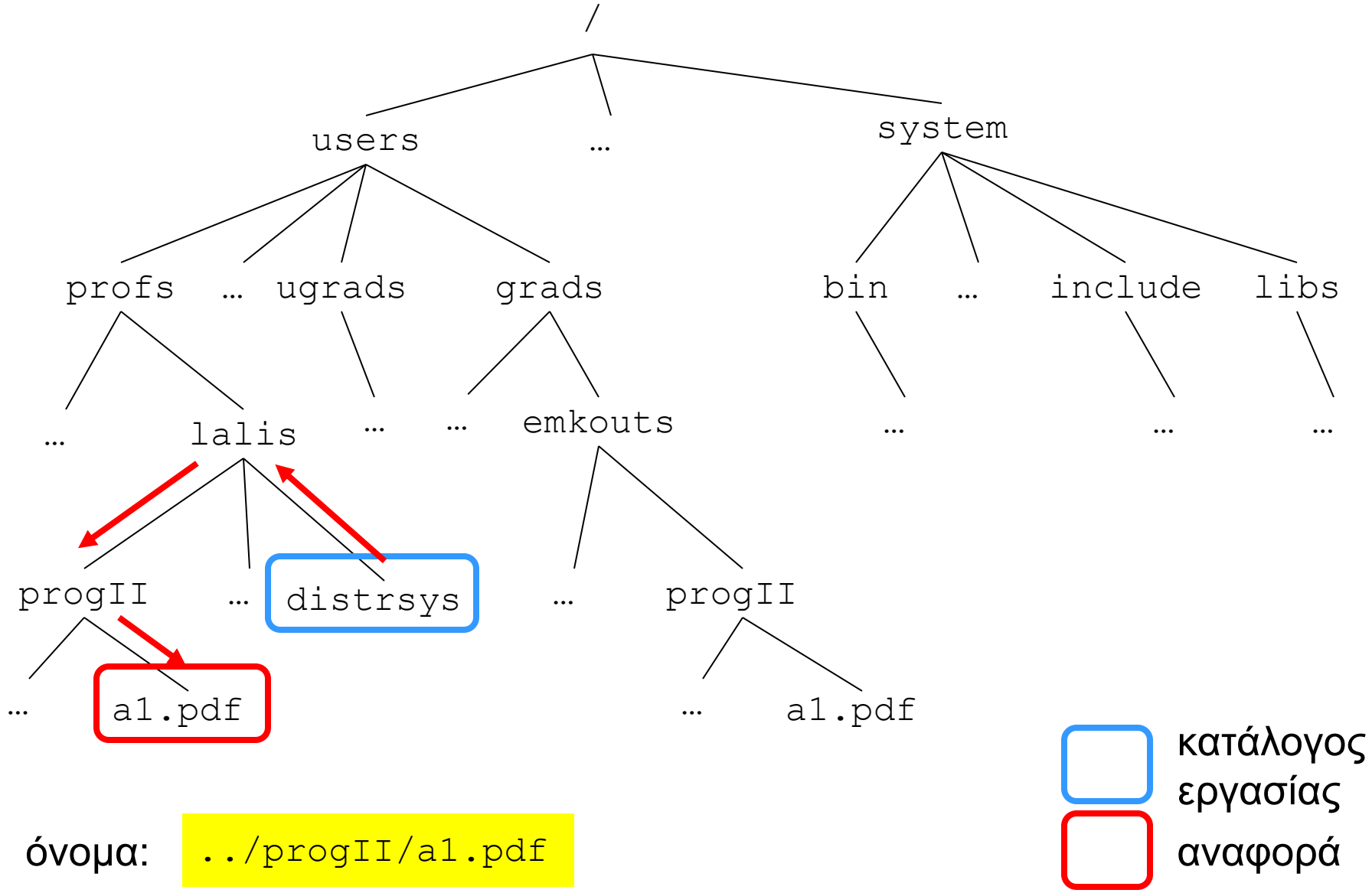
όνομα: `progII/a1.pdf`



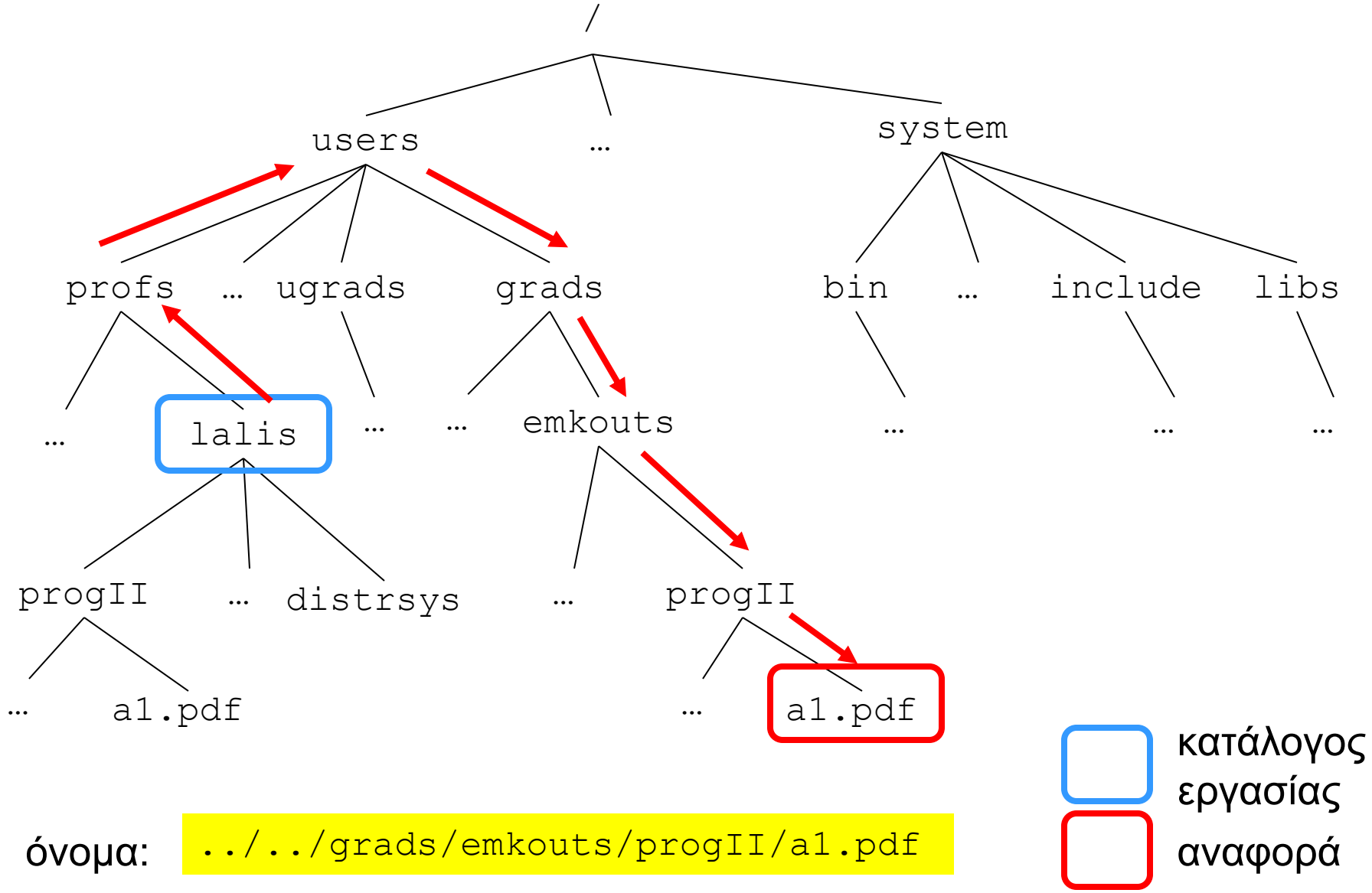
όνομα: `/users/profs/lalis/progII/a1.pdf`



όνομα: `/users/profs/lalis/progII/a1.pdf`



όνομα: `../progII/a1.pdf`



όνομα: `../../grads/emkouts/progII/a1.pdf`

Προστασία αρχείων / καταλόγων

- Για κάθε αρχείο / κατάλογο το λειτουργικό διατηρεί πληροφορία για τον **ιδιοκτήτη** του, καθώς και για τις **άδειες πρόσβασης** που αυτός έχει δώσει
- Υποκείμενα αδειών: **user**, **group**, **others**
- Ξεχωριστές άδειες για κάθε υποκείμενο
- Όταν ένα πρόγραμμα επιχειρεί να προσπελάσει ένα αρχείο, το λειτουργικό **ελέγχει** την άδεια πρόσβασης
  - αν ο χρήστης στο όνομα του οποίου εκτελείται το πρόγραμμα δεν έχει άδεια να προσπελάσει το αρχείο, η λειτουργία δεν εκτελείται
- Αλλαγές ιδιοκτησίας γίνονται μέσω `chown` / `fchown`
- Αλλαγές αδειών γίνονται μέσω `chmod` / `fchmod`

Προσδιορισμός αδειών με `chmod`

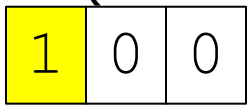
- Προσδιορισμός **υποκειμένων**
  - `u(ser)` χρήστης/ιδιοκτήτης
  - `g(roup)` ομάδα
  - `o(thers)` άλλοι
  - `a(ll)` όλοι
- Προσδιορισμός **άδειας**
  - `r(ead)` ανάγνωση
  - `w(rite)` γράψιμο/αλλαγή
  - `(e)x(ecute)` εκτέλεση

- `chmod a+rwx myfile`
- `chmod g-x,o-wx myfile`

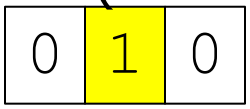
Οκταδική κωδικοποίηση αδειών

- Προσδιορισμός άδειας με 3 bits (1 οκταδικό ψηφίο)

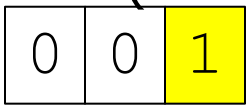
read (octal `4`)



write (octal `2`)

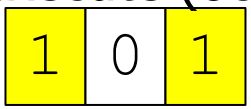


execute (octal `1`)

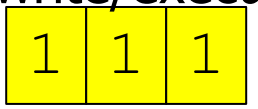


- Οι άδειες μπορεί να συνδυαστούν μεταξύ τους

read/execute (octal `5`)

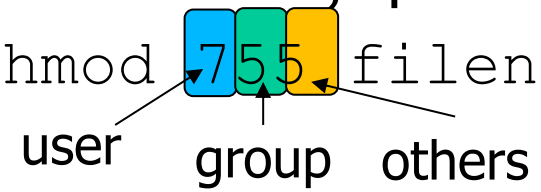


read/write/execute (octal `7`)



- Συνολικά, χρησιμοποιούνται 3 οκταδικά ψηφία
  - για χρήστη (user), ομάδα (group), άλλους (others)

- Εναλλακτικός τρόπος προσθαφαίρεσης αδειών
- `chmod 755 filename`



Περιορισμός αδειών μέσω `umask`

- Ο χρήστης μπορεί να **μπλοκάρει** συγκεκριμένες άδειες για τα αρχεία που δημιουργούνται από τα προγράμματα που εκτελεί, μέσω της μάσκας `umask`
- Η μάσκα ορίζεται μέσω της εντολής/λειτουργίας `umask`

- Κατά την εκτέλεση, γίνεται συνδυασμός των αδειών `perms` που δίνει ένα πρόγραμμα (μέσω της λειτουργίας `open`) με την μάσκα του χρήστη `perms & ~umask`

- `umask=077` μπλοκάρει όλες τις άδειες για `g+o`
- `umask=022` μπλοκάρει την άδεια γραψίματος για `g+o`

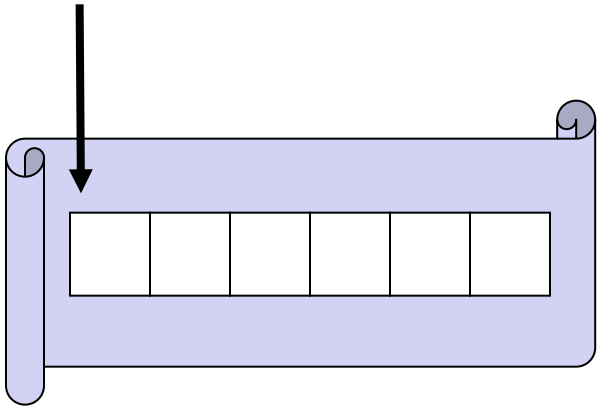
Προσπέλαση περιεχομένων αρχείου

- Τα bytes γράφονται/διαβάζονται **σειριακά**

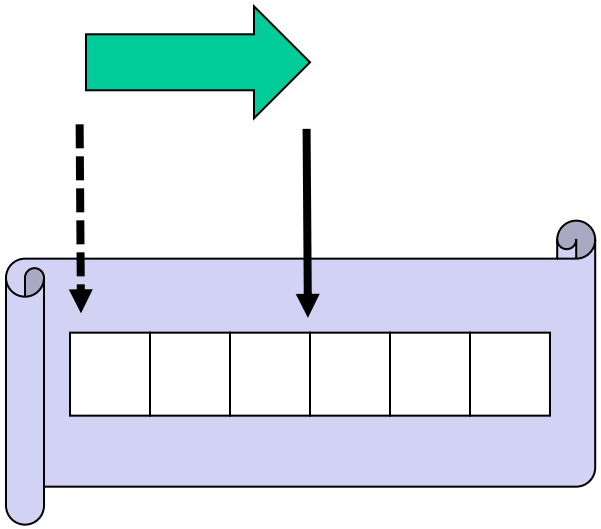
- Η θέση εγγραφής/ανάγνωσης **αυξάνεται** αυτόματα μέσα από τις λειτουργίες ανάγνωσης ή γραψίματος
- Όταν διαβάζονται bytes
  - **μόνο** όσο δεν φτάνουμε στο τέλος του αρχείου
- Όταν γράφονται bytes
  - τα bytes που γράφονται **αντικαθιστούν** τα υπάρχοντα περιεχόμενα που τυχόν βρίσκονται στις αντίστοιχες θέσεις

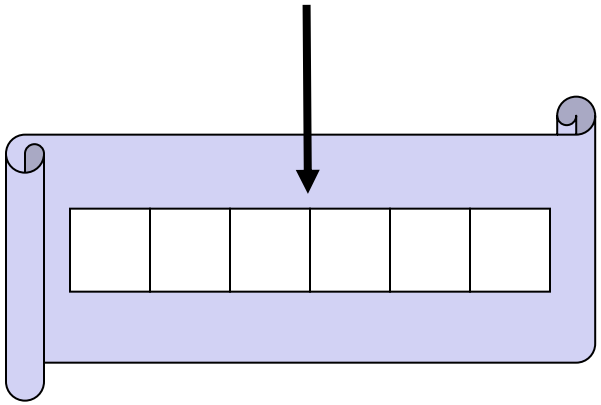
- Το αρχείο **επεκτείνεται/μεγαλώνει** αυτόματα όταν γράφονται bytes πέρα από το τέλος του

θέση ανάγνωσης / εγγραφής μετά το **άνοιγμα**

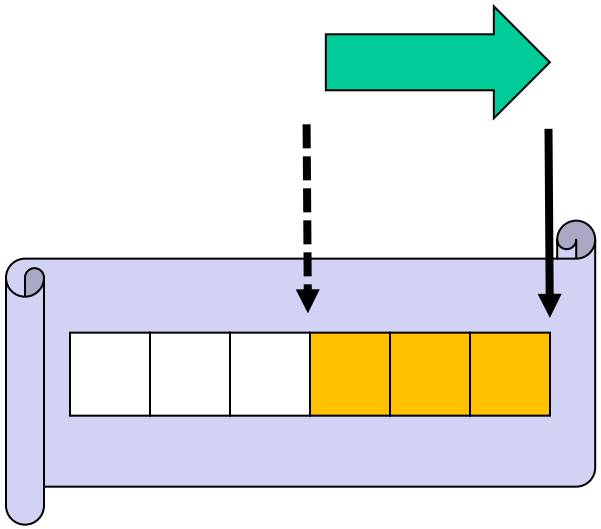


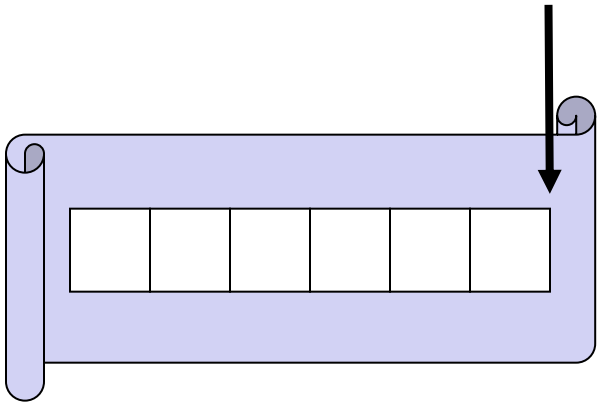
θέση ανάγνωσης / εγγραφής μετά από **διάβασμα** 3 bytes



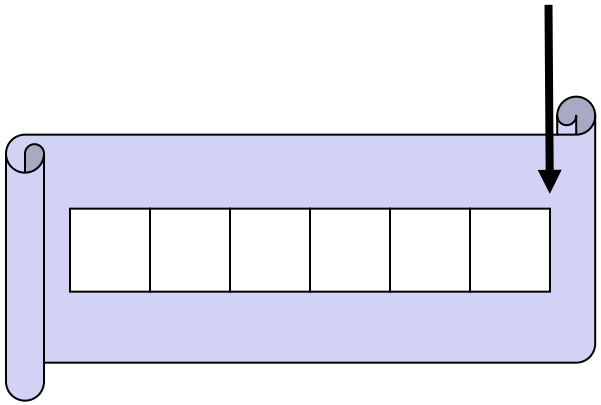


θέση ανάγνωσης / εγγραφής μετά από **γράψιμο** 3 bytes

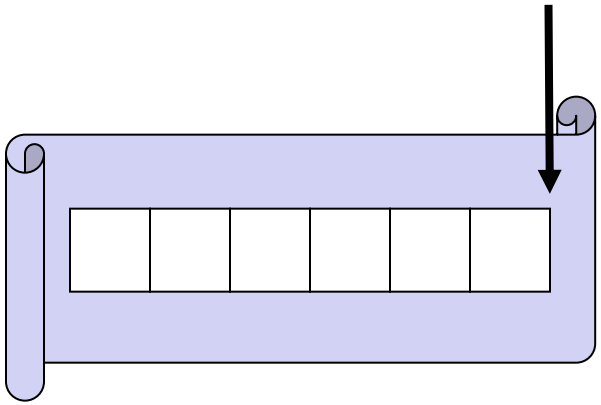




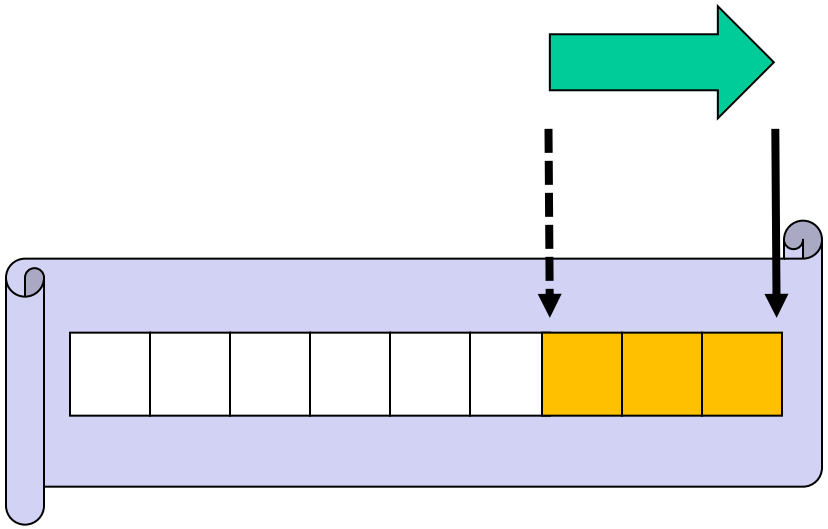
θέση ανάγνωσης / εγγραφής μετά από **διάβασμα** 3 bytes



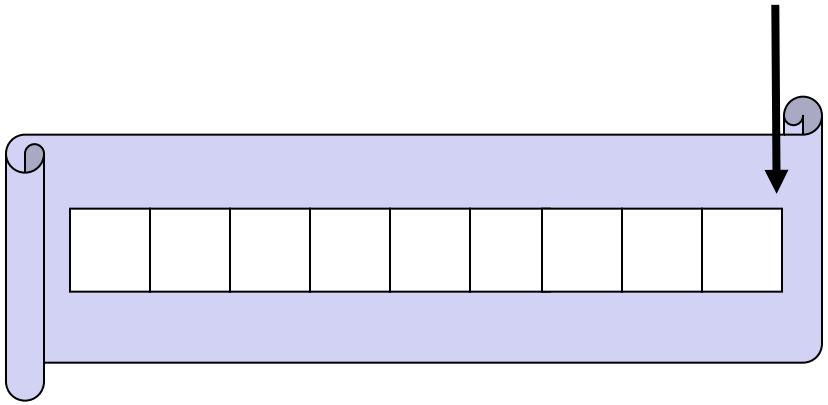
δεν διαβάστηκε **τίποτα**
(αφού έχουμε ήδη φτάσει στο τέλος του αρχείου)



θέση ανάγνωσης / εγγραφής μετά από **γράψιμο** 3 bytes



το αρχείο **επεκτάθηκε** αυτόματα

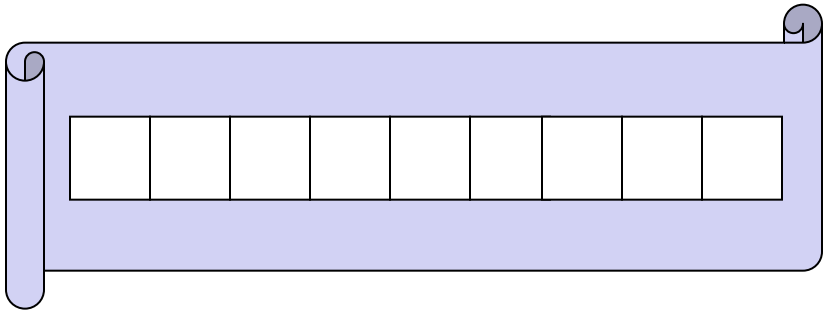


Κόψιμο (και επέκταση) αρχείου

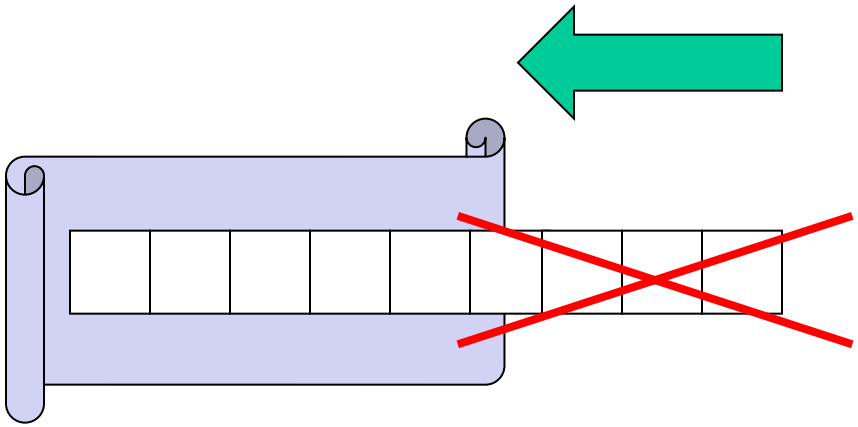
- **Δεν** υπάρχει λειτουργία για την απομάκρυνση bytes από την αρχή ή από το μέσο του αρχείου

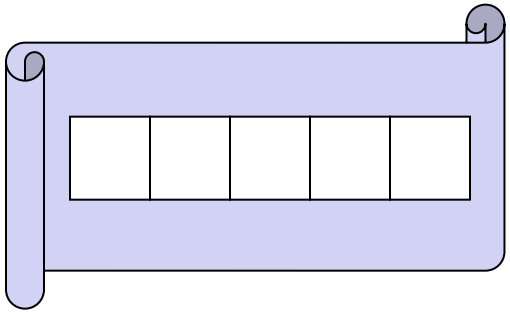
- Το αρχείο μπορεί να **κοπεί** (από το τέλος)
- Υπάρχει **ειδική λειτουργία** για αυτό

- Με την ίδια λειτουργία, το αρχείο μπορεί να **επεκταθεί** (να μεγαλώσει σε μέγεθος)
  - **χωρίς** να γράψει δεδομένα η εφαρμογή

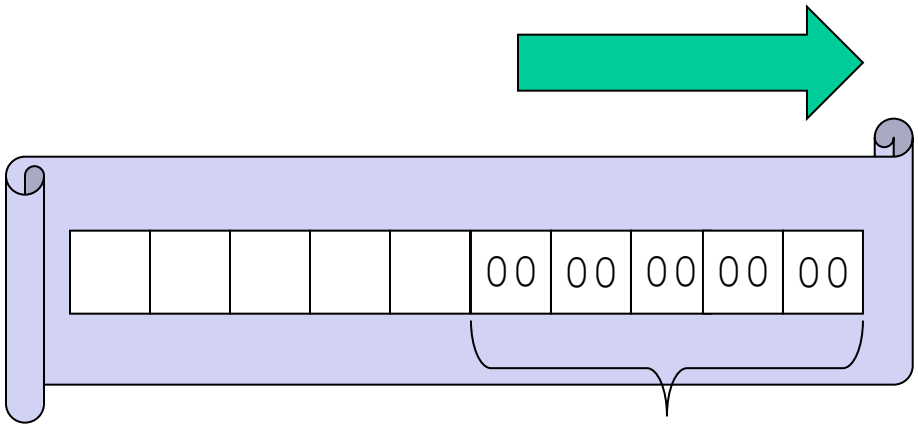


κόψιμο αρχείου στα 5 bytes





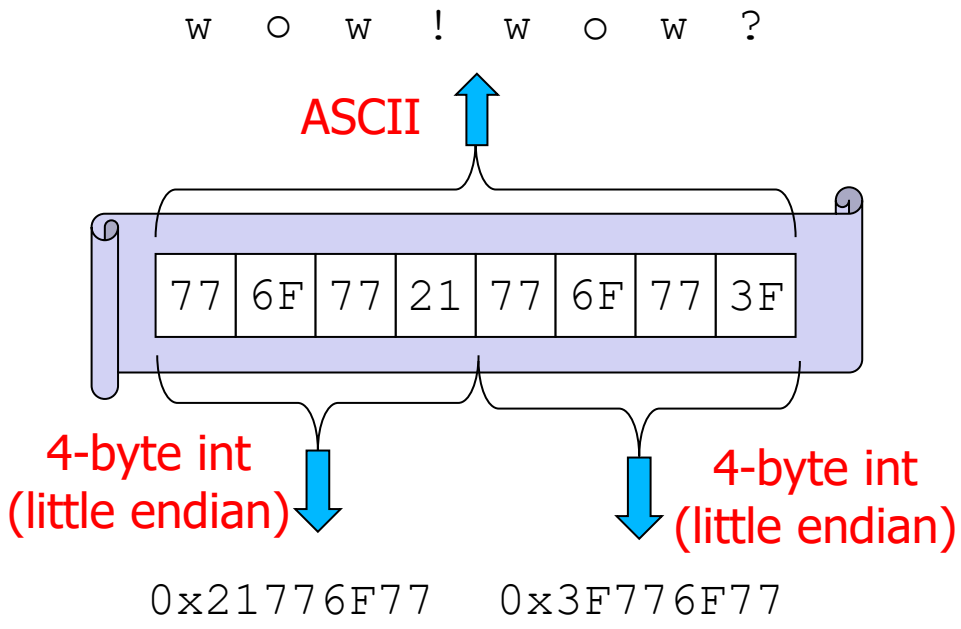
επέκταση αρχείου στα 10 bytes



τα επιπλέον bytes
αρχικοποιούνται
αυτομάτως σε 0

Ερμηνεία περιεχομένων ενός αρχείου

- **Text file:** το περιεχόμενο του αρχείου μπορεί να ερμηνευθεί με βάση την κωδικοποίηση ASCII
- **Binary file:** ένα αρχείο που δεν είναι (μόνο) text
- Τα περιεχόμενα ενός αρχείου (ιδίως αν είναι binary) **δεν** είναι «προφανώς» ερμηνεύσιμα
  - χρειάζονται **συμβάσεις** κωδικοποίησης/ερμηνείας των δεδομένων
  - ακόμα και τα περιεχόμενα ενός text file μπορεί να «επιδέχονται» πολλές και διαφορετικές ερμηνείες – γιατί;
- Το λειτουργικό σύστημα απλά διαβάζει/γράφει bytes
  - **χωρίς** να κάνει καμία προσπάθεια να τα ερμηνεύσει
- Η **κωδικοποίηση/ερμηνεία** των περιεχομένων ενός αρχείου είναι **ευθύνη των εφαρμογών**


w o w ! w o w ?
ASCII
77 6F 77 21 77 6F 77 3F
4-byte int
(little endian)
4-byte int
(little endian)
0x21776F77
0x3F776F77

Τύποι αρχείων

- Το λειτουργικό **δεν** διατηρεί πληροφορία «τύπου» για τα αρχεία που δημιουργούν τα προγράμματα
  - ούτε ενδιαφέρεται να ερμηνεύσει τα περιεχόμενα τους
- Πληροφορία «τύπου» προστίθεται στο αρχείο, **κωδικοποιημένη μέσα στα περιεχόμενα** του
  - από το πρόγραμμα που δημιουργεί το αρχείο

- Η κατάληξη ενός αρχείου (π.χ., `.pdf`) είναι απλά μια **σύμβαση** για να γίνεται πιο εύκολη αναζήτηση
  - όταν ο χρήστης ψάχνει για κάποιο αρχείο
  - είναι απλά ενδεικτική για τα περιεχόμενα του αρχείου (π.χ. αρχεία με κατάληξη .pdf **δεν** είναι εγγυημένα pdf)

Τέλος αρχείου

- Το τέλος των αρχείων **δεν** κωδικοποιείται μέσα στο ίδιο το αρχείο, π.χ. μέσω κάποιου «ειδικού» byte
  - οποιοδήποτε byte ή σειρά bytes μπορεί να βρίσκεται μέσα στο «κανονικό» περιεχόμενο του αρχείου

- Το τέλος του αρχείου είναι αποθηκευμένο σε μια **ξεχωριστή δομή δεδομένων** που διαχειρίζεται εσωτερικά το λειτουργικό σύστημα

- Αυτή η δομή χρησιμοποιείται για να αποθηκευτούν και άλλες πληροφορίες σχετικές με το αρχείο

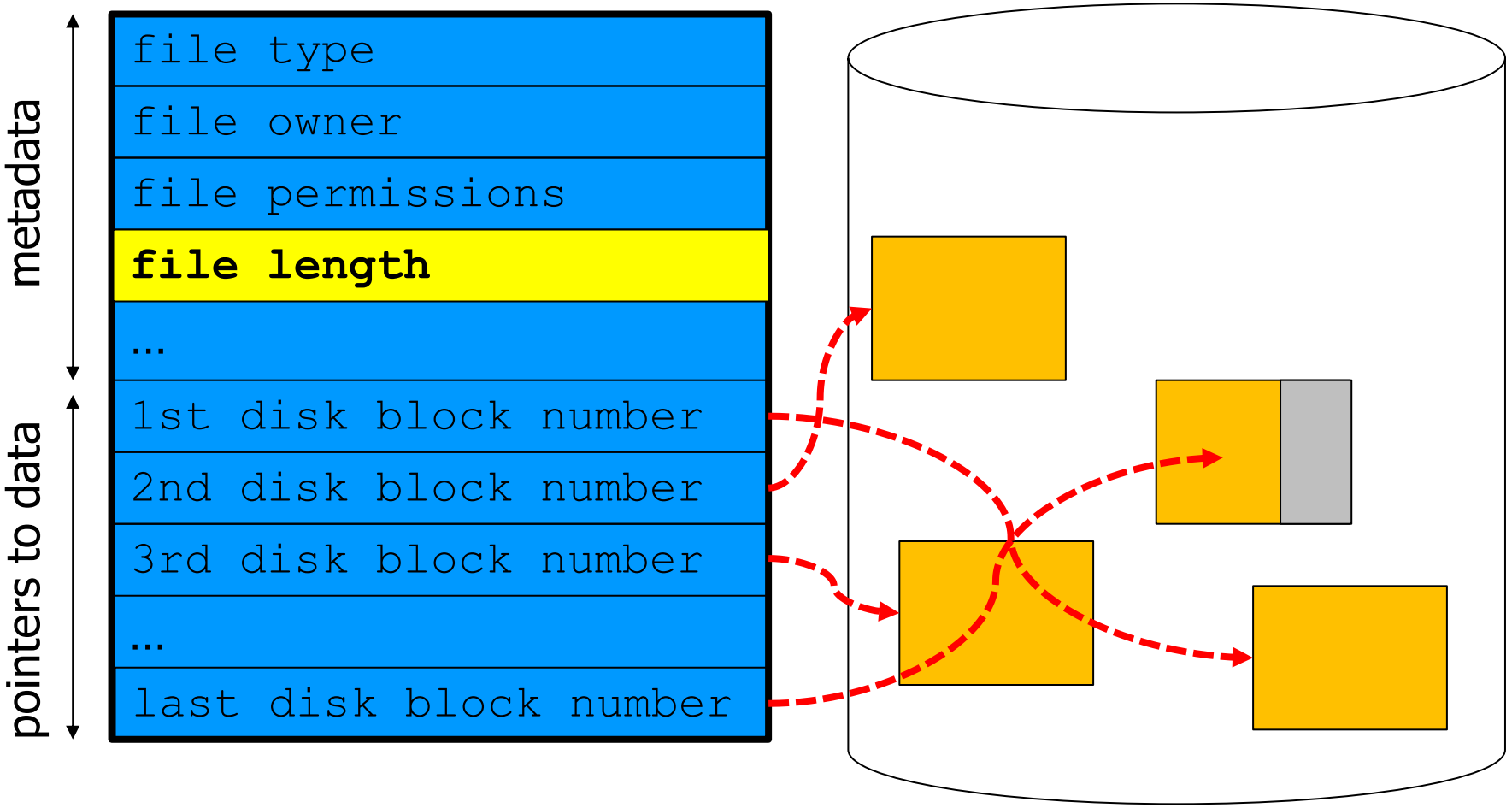
Δομή/κόμβος πληροφορίας αρχείου

- Για κάθε αρχείο, το λειτουργικό διατηρεί μια ειδική δομή, το λεγόμενο **information node** ή **inode**

- Το inode περιέχει διάφορες χρήσιμες πληροφορίες για το αρχείο
  - τύπος αρχείου, ιδιοκτήτης, ομάδα, άδειες πρόσβασης
  - αριθμός συσκευής που βρίσκεται το αρχείο
  - χρονοσφραγίδα πρόσφατης πρόσβασης/αλλαγής
  - **μέγεθος**, αριθμός μπλοκ
  - δείκτες στα μπλοκ όπου βρίσκονται αποθηκευμένα τα δεδομένα

- Για κάθε αρχείο, υπάρχει **μόνο ένα** inode

file information structure



Κατάλογοι

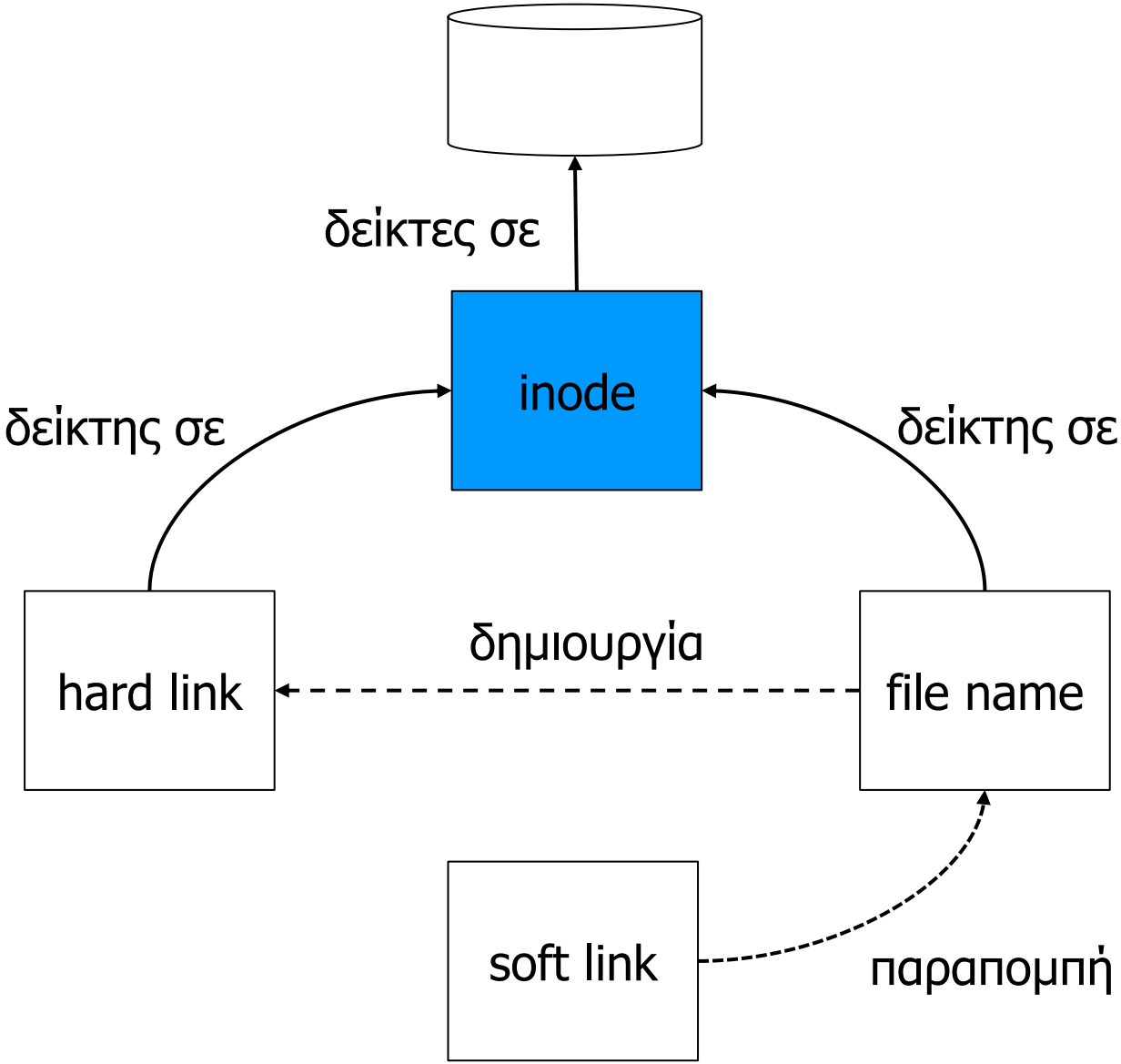
- Ο κατάλογος είναι πρακτικά ένα **ευρετήριο** αρχείων

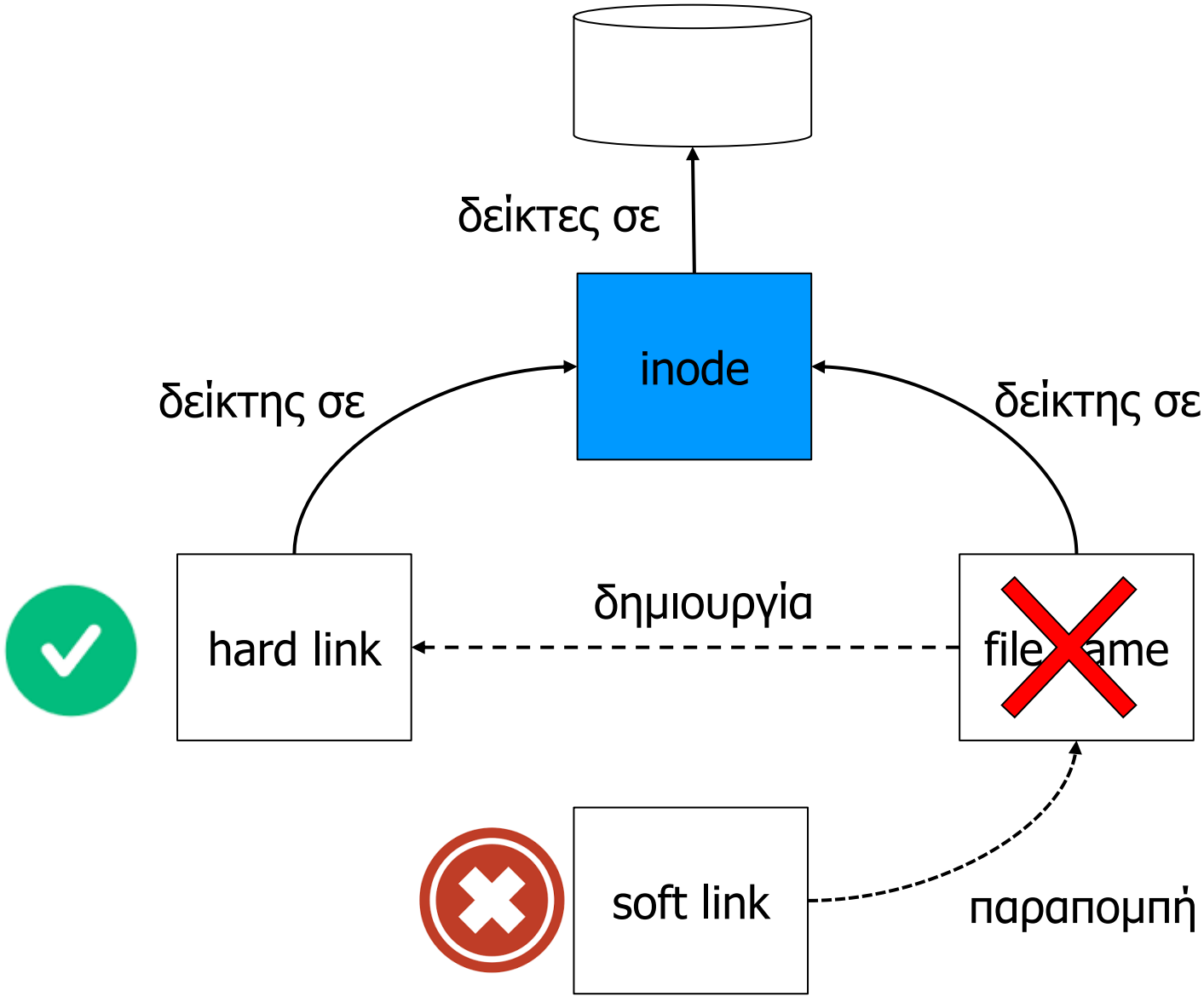
- Συνήθως, υλοποιούνται ως **ειδικά** αρχεία
- Τα περιεχόμενα των οποίων είναι τα **ονόματα** των αρχείων του καταλόγου καθώς και αναφορές στις αντίστοιχες δομές πληροφορίας (i-node numbers)

- Κλασικές αλλά λίγο **παραπλανητικές** εκφράσεις
  - «ο κατάλογος ***περιέχει*** αρχεία»
  - «το αρχείο βρίσκεται ***μέσα*** στον κατάλογο»

Αρχεία, ονόματα αρχείων, σύνδεσμοι

- Το «αρχείο» και το «όνομα αρχείου» δεν είναι ταυτόσημες έννοιες
- **Αρχείο (file):** περιέχει τα δεδομένα
  - αντιστοιχεί σε ένα μοναδικό i-node
- **Όνομα αρχείου (file name):** αναφορά μέσω της οποίας μπορούμε να εντοπίσουμε ένα αρχείο
  - στο i-node του αρχείου
- **Σύνδεσμος (link):** ένα (εναλλακτικό) όνομα για ένα ήδη υπάρχον αρχείο
  - hard links: ονόματα που δείχνουν σε δομή πληροφορίας
  - soft links: ειδικά (μικρά) αρχεία που έχουν ως περιεχόμενο το όνομα ενός αρχείου


δείκτες σε
inode
δείκτης σε
δείκτης σε
hard link
δημιουργία
file name
soft link
παραπομπή


δείκτες σε
inode
δείκτης σε
δείκτης σε
hard link
δημιουργία
file name
soft link
παραπομπή

Παράδειγμα

- Δημιουργία αρχείου

```
$ echo hello > test
$ ls -il test*
```

- Δημιουργία hard link

```
$ ln test testlink
$ ls -il test*
```

- Απομάκρυνση πρώτου/αρχικού ονόματος

```
$ rm test
$ ls -il test*
```