Αρχεία

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)

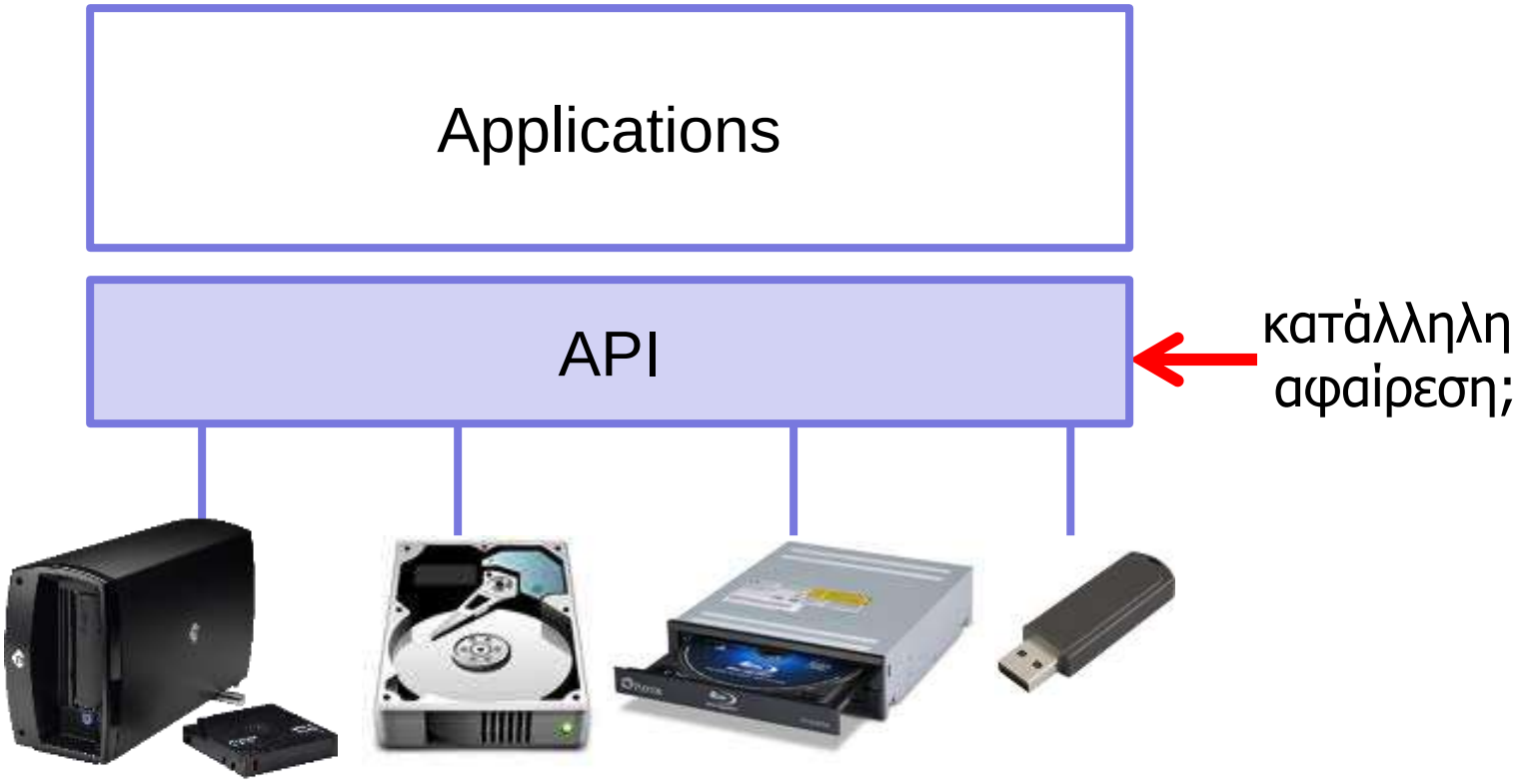
- Χρειαζόμαστε **μόνιμα** αποθηκευτικά μέσα που λειτουργούν **χωρίς** παροχή ρεύματος
  - μαγνητικά, οπτικά, solid-state, …

- 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
API
κατάλληλη αφαίρεση;

Αρχεία

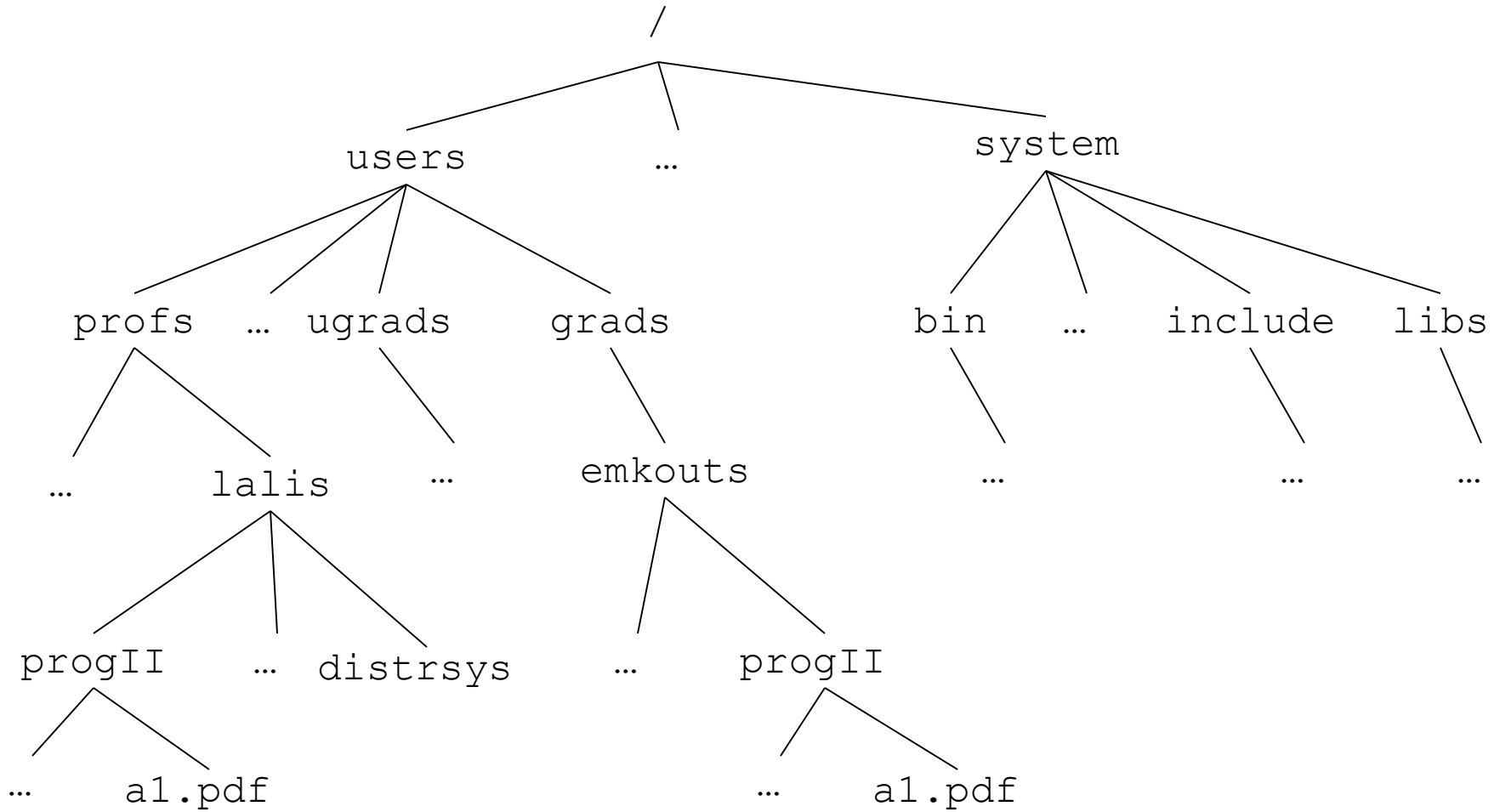
- Ξεχωριστές **οντότητες αποθήκευσης** δεδομένων

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

- Τα αρχεία είναι **επώνυμα**: κάθε αρχείο έχει «μοναδικό» **όνομα/αναγνωριστικό**
  - για να μπορεί να **εντοπιστεί** αργότερα, είτε από το πρόγραμμα που το δημιούργησε είτε από άλλα προγράμματα, είτε από τον ίδιο τον χρήστη

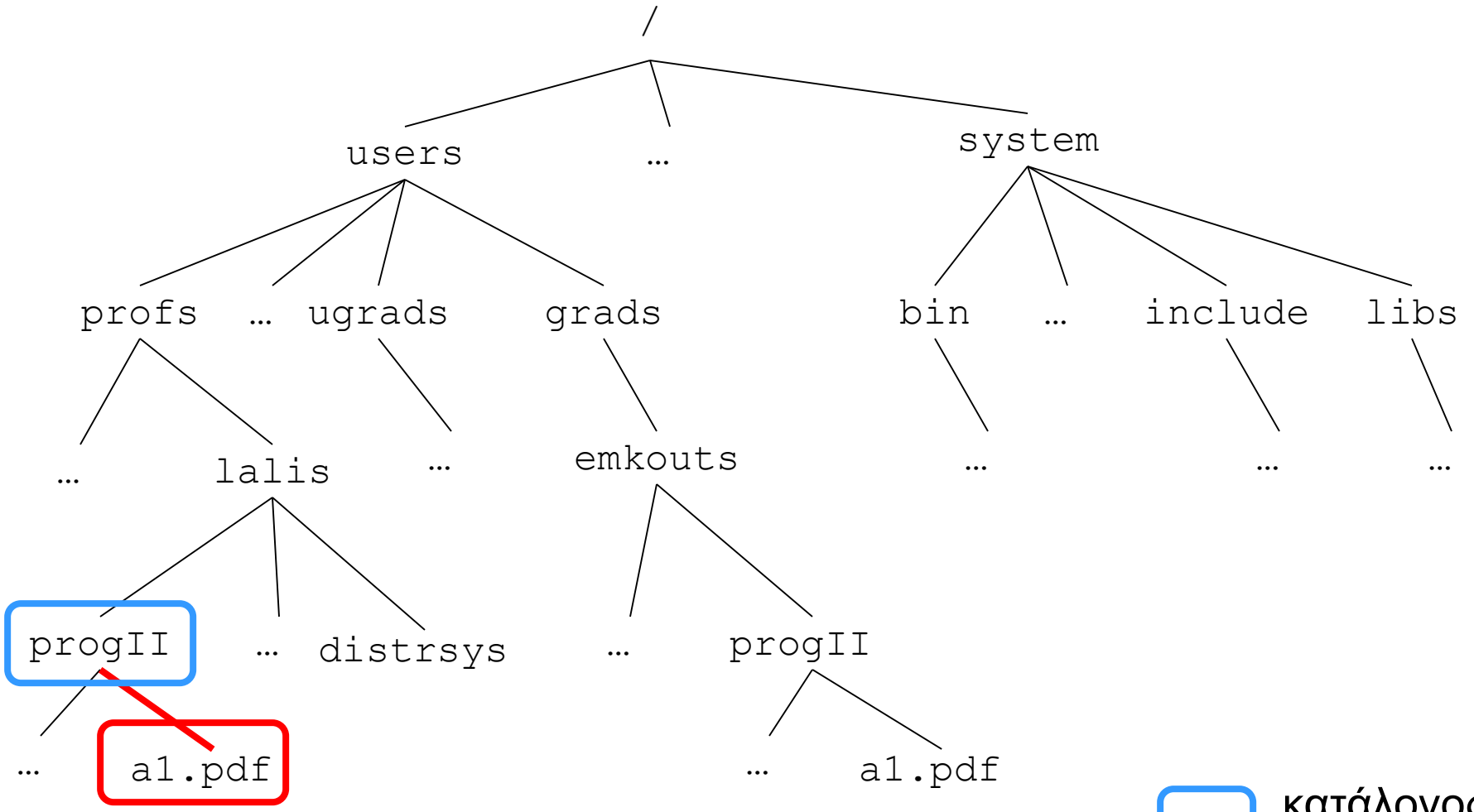
Χώρος ονομάτων (name space)

- Τα ονόματα αρχείων δίνονται στο πλαίσιο ενός **«χώρου ονομάτων»** του συστήματος
- Ο χώρος είναι δομημένος **ιεραρχικά**
  - διευκολύνει την ανάθεση μοναδικών ονομάτων
  - 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

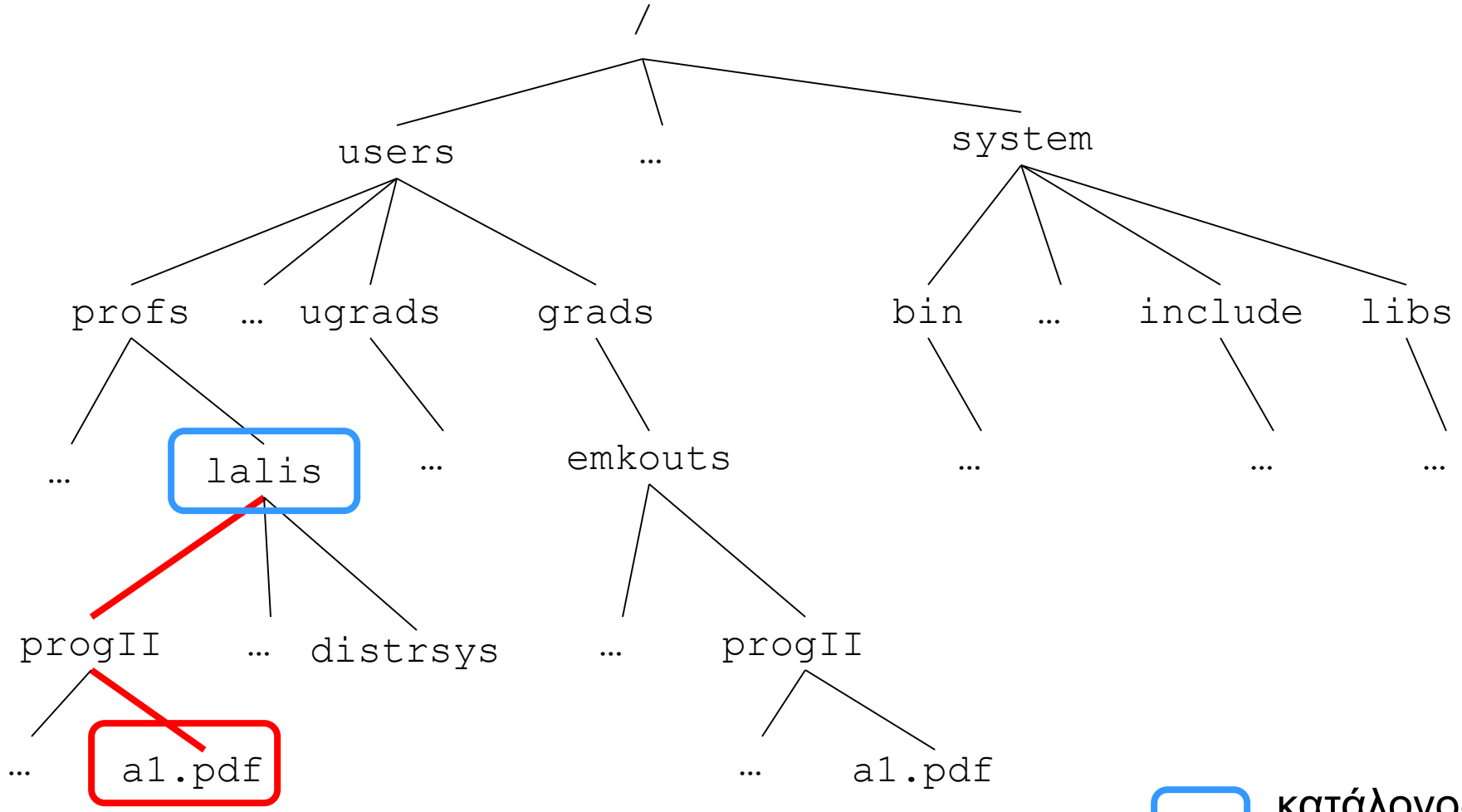
- **Απόλυτος** προσδιορισμός, σε **καθολικό** επίπεδο
  - πλήρες όνομα, με βάση την **κορυφή της ιεραρχίας**
- **Σχετικός** προσδιορισμός, σε **τοπικό** επίπεδο
  - τμήμα ονόματος, με βάση τον **κατάλογο εργασίας**
- Συμβάσεις
  - "." για τον τρέχοντα κατάλογο εργασίας
  - ".." για τον κατάλογο που βρίσκεται ένα επίπεδο ψηλότερα από τον τρέχοντα κατάλογο εργασίας
  - "/" για την ρίζα του δέντρου, αλλά και ως διαχωριστικό ανάμεσα στα ονόματα καταλόγων/αρχείων ενός μονοπατιού



όνομα: a1.pdf

κατάλογος εργασίας

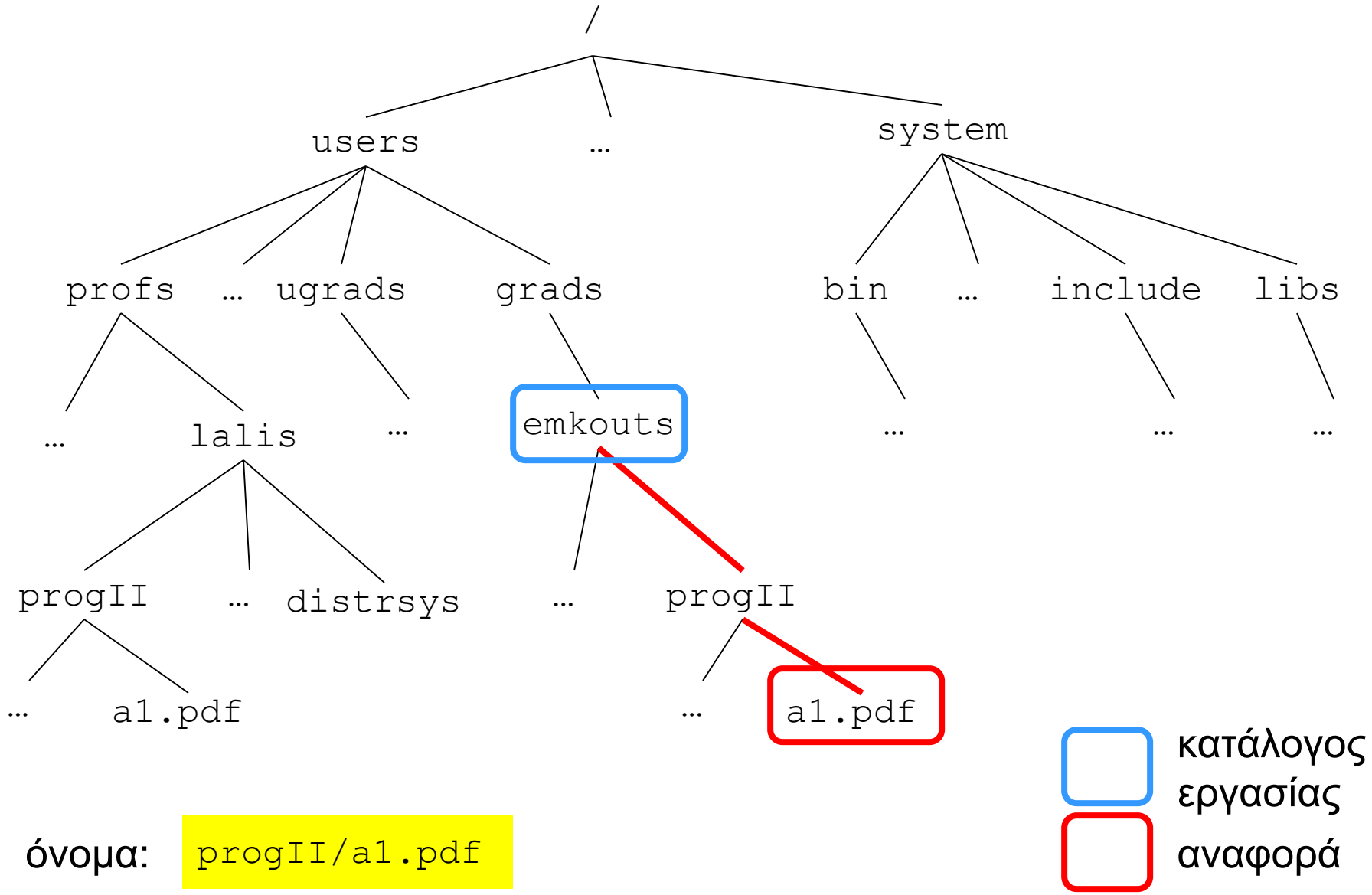
αναφορά



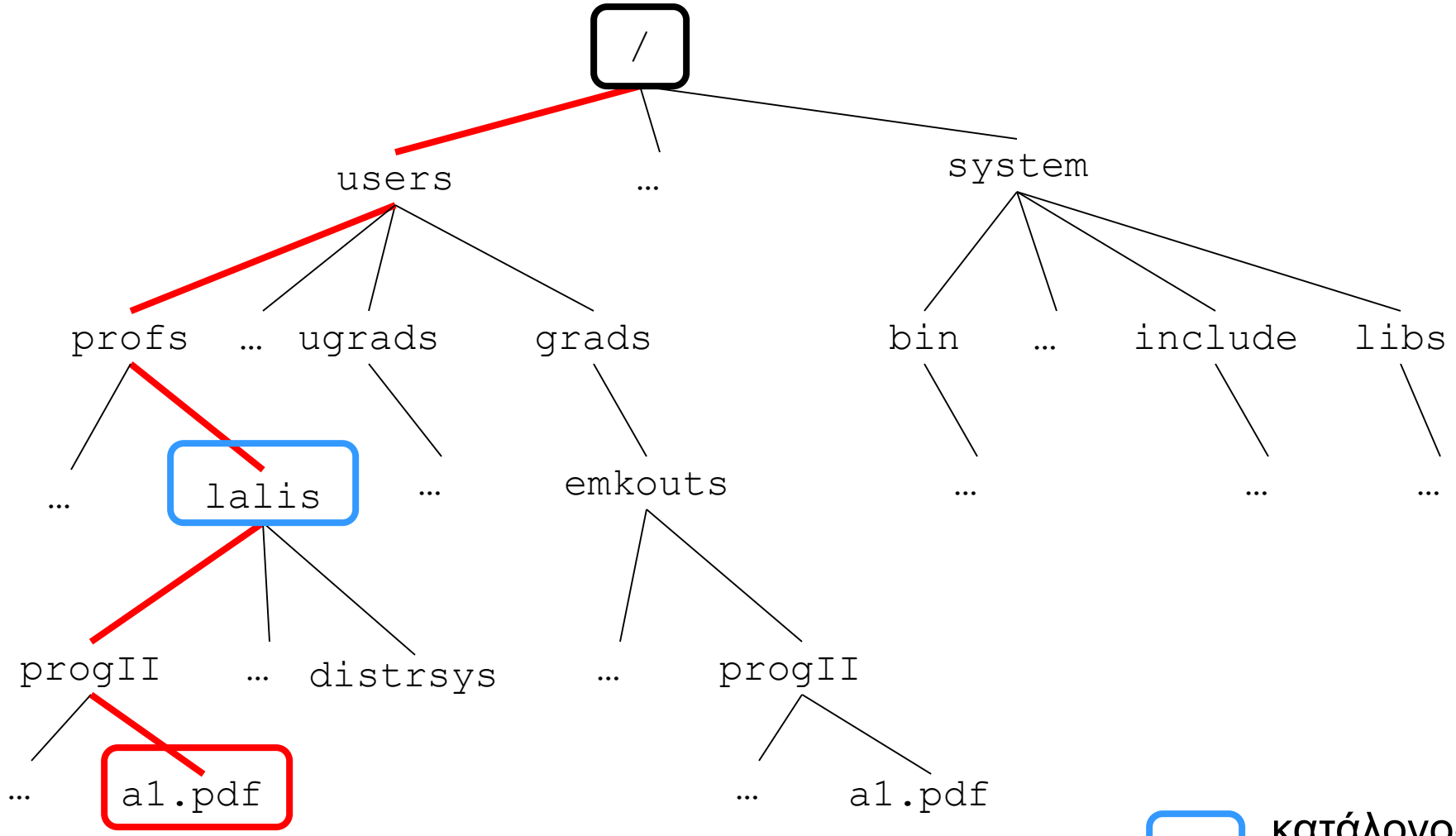
όνομα: `progII/a1.pdf`

κατάλογος εργασίας

αναφορά



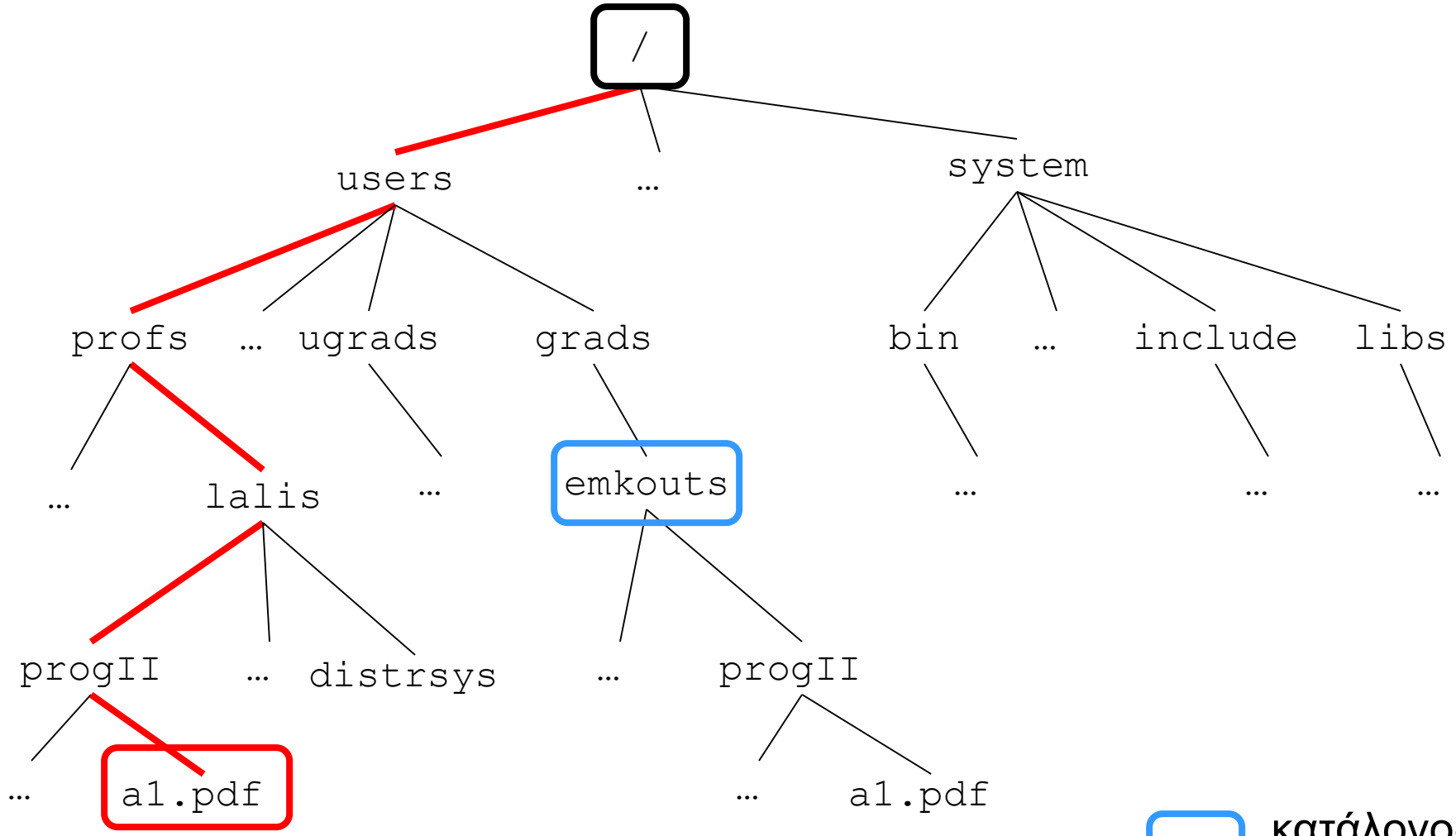
όνομα: `progII/a1.pdf`



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

κατάλογος εργασίας

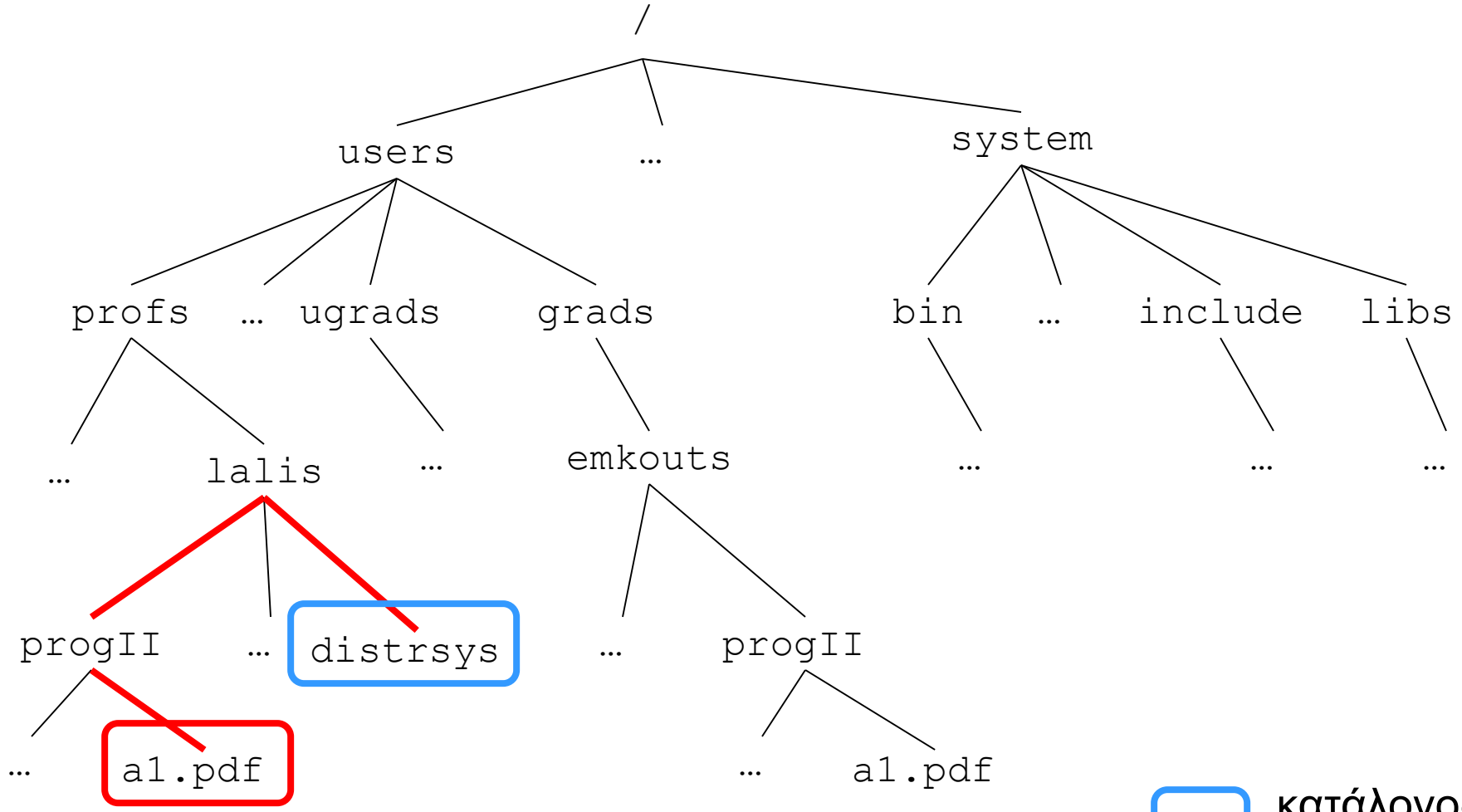
αναφορά



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

κατάλογος εργασίας

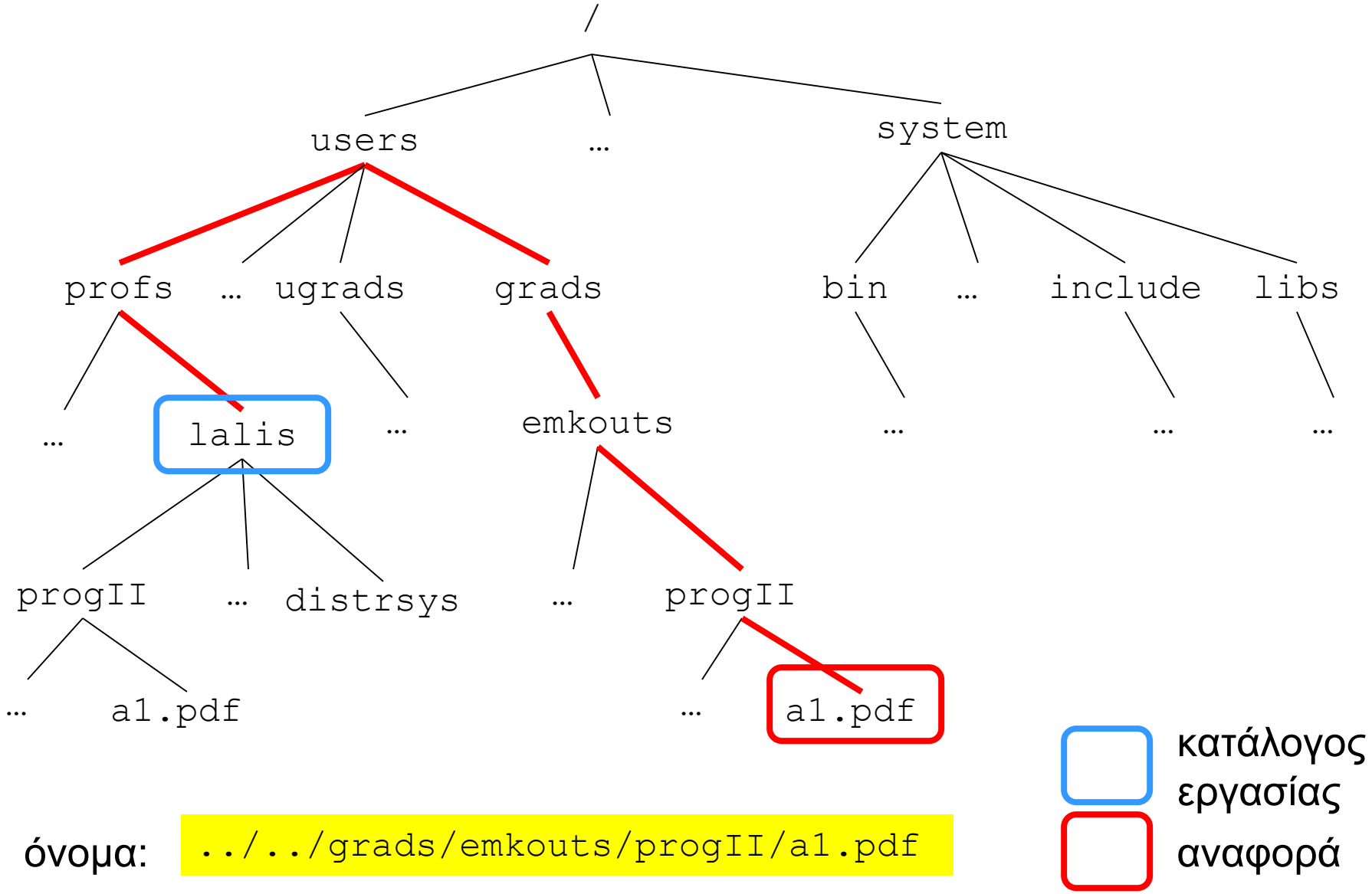
αναφορά



κατάλογος εργασίας

αναφορά

όνομα: `../progII/a1.pdf`



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

Προστασία αρχείων / καταλόγων

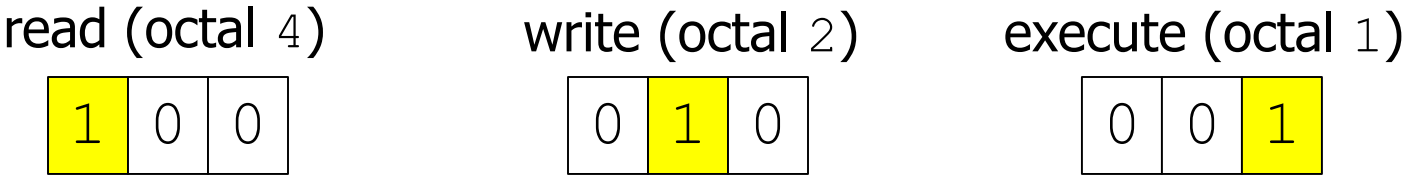
- Για κάθε αρχείο / κατάλογο το λειτουργικό διατηρεί πληροφορία για τον **ιδιοκτήτη** του, καθώς και για τις **άδειες πρόσβασης** στο αρχείο
- Υποκείμενα αδειών: **user**, **group**, **others**
- Ξεχωριστές άδειες για κάθε υποκείμενο
- Όταν ένα πρόγραμμα επιχειρεί να προσπελάσει ένα αρχείο, το λειτουργικό **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`

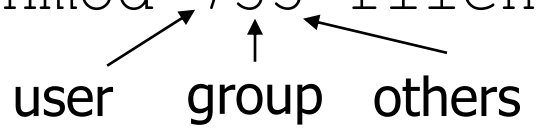
- Προσδιορισμός υποκειμένων
  - `u(ser)` χρήστης/ιδιοκτήτης
  - `g(group)` ομάδα
  - `o(thers)` άλλοι
  - `a(ll)` όλοι
- Προσδιορισμός άδειας
  - `r(ead)` ανάγνωση
  - `w(rite)` γράψιμο/αλλαγή
  - `(e)x(ecute)` εκτέλεση

- `chmod u+rwx filename`
- `chmod o+rx,o-w filename`

Οκταδική κωδικοποίηση αδειών



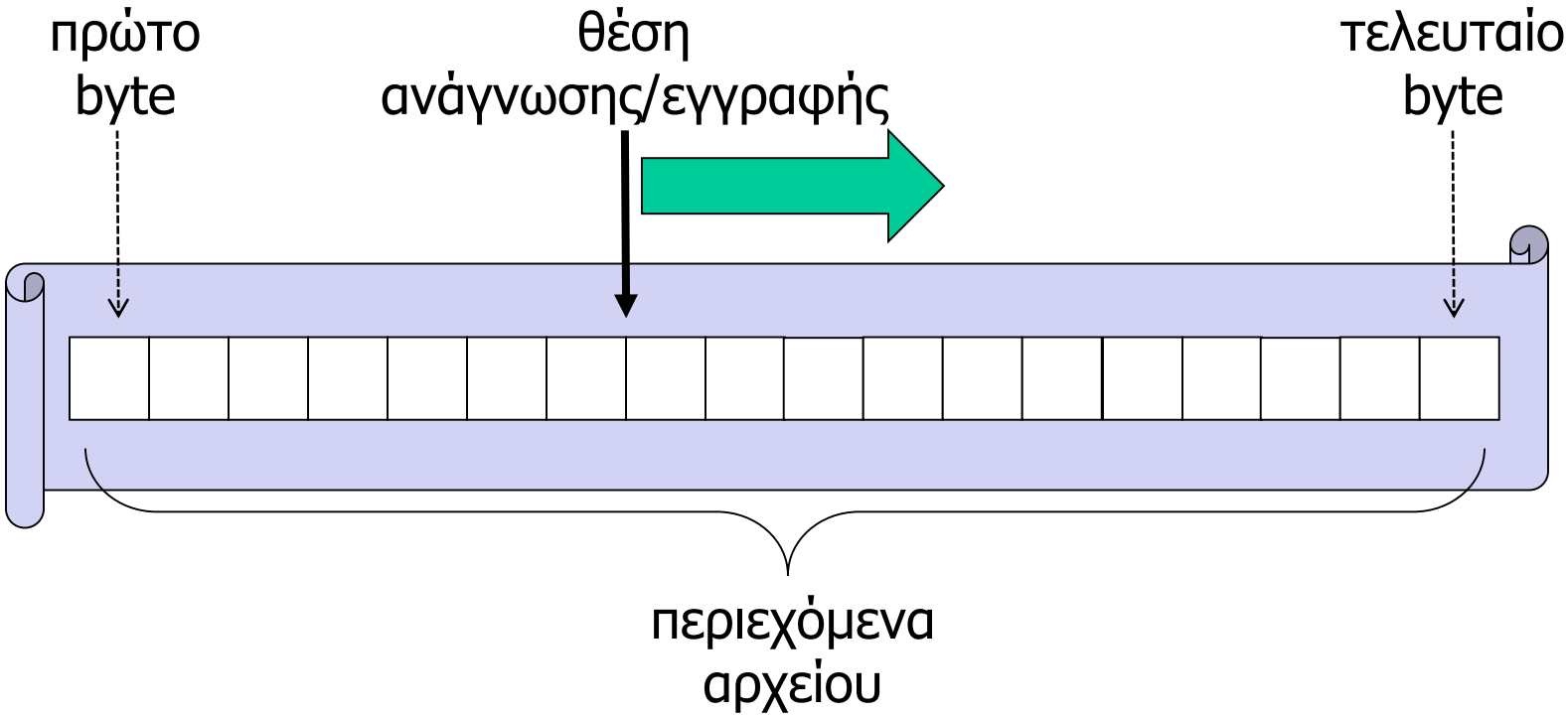
- Προσδιορισμός άδειας
  - `4` ανάγνωση
  - `2` γράψιμο/αλλαγή
  - `1` εκτέλεση
- Ξεχωριστά οκταδικά ψηφία για χρήστη-ομάδα-άλλους
  - με αυτή την σειρά
- Πιο εύκολη προσθαφαίρεση αδειών
- `chmod 755 filename`



Γράψιμο/διάβασμα δεδομένων σε/από αρχείο

- Τα bytes γράφονται/διαβάζονται **σειριακά**
- Η θέση εγγραφής/ανάγνωσης **αυξάνεται αυτόματα** όταν γράφονται/διαβάζονται bytes (κατά τον αριθμό των bytes που γράφτηκαν/διαβάστηκαν)
- Η θέση εγγραφής/ανάγνωσης μπορεί να αλλάξει (προς τα εμπρός ή πίσω) και με **ρητό** τρόπο

- Το αρχείο **επεκτείνεται/μεγαλώνει αυτόματα**, όταν γράφονται bytes πέρα από το τέλος του
- Το αρχείο μπορεί να **συρρικνωθεί/κοπεί**, από το τέλος προς την αρχή – αυτό πρέπει να γίνει **ρητά**
  - **δεν** υπάρχει λειτουργία για το σβήσιμο/αποκοπή περιεχομένων από την αρχή ή το μέσο του αρχείου


πρώτο
byte
θέση
ανάγνωσης/εγγραφής
τελευταίο
byte
περιεχόμενα
αρχείου

Ερμηνεία περιεχομένων ενός αρχείου

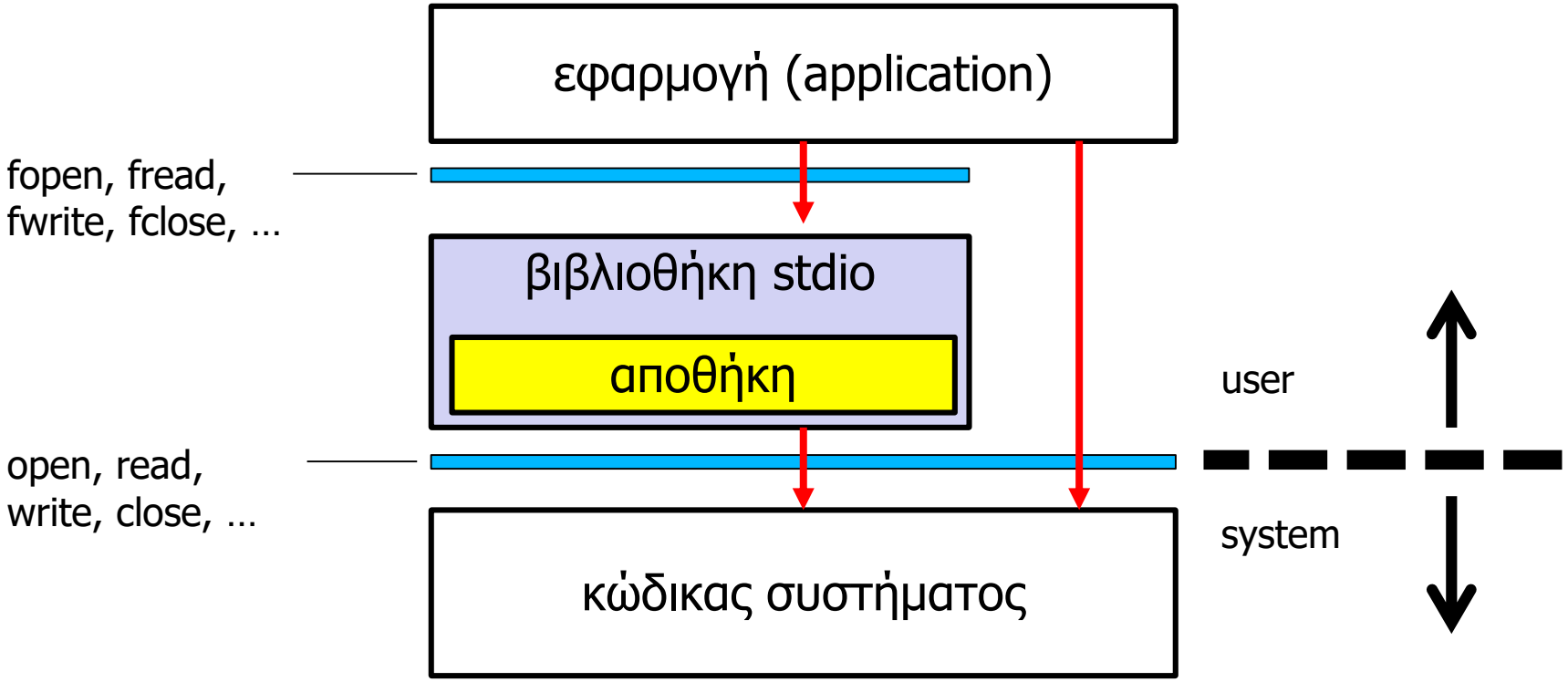
- Το λειτουργικό σύστημα διαβάζει/γράφει bytes από/σε αρχεία **χωρίς** να τα ερμηνεύει
- Η **σημασία/ερμηνεία** των περιεχομένων ενός αρχείου είναι **ευθύνη του προγράμματος**
- Χρειάζονται κατάλληλες **συμβάσεις/πρότυπα** κωδικοποίησης/ερμηνείας των δεδομένων

- Η τιμή ενός ή πολλών bytes μπορεί να ερμηνευθεί με **πολλούς διαφορετικούς** τρόπους
- Text file: το περιεχόμενο του αρχείου πρέπει να ερμηνευθεί χρησιμοποιώντας την σύμβαση ASCII
- Binary file: ένα αρχείο που δεν είναι (μόνο) text

Η βιβλιοθήκη stdio

- Ορίστηκε από τον Dennis Ritchie το 1975
- Μέρος του προτύπου ANSI C

- Λειτουργίες υψηλού επιπέδου για ανάγνωση και γράψιμο δεδομένων, **ανεξάρτητες** του εκάστοτε λειτουργικού συστήματος
- Διάβασμα/γράψιμο από/σε **ροές δεδομένων** (streams)
- Υποστήριξη για μορφοποιημένη εκτύπωση/σκανάρισμα συμβολοσειρών (strings) και αριθμητικών τιμών
- Χρησιμοποιεί εσωτερική αποθήκη δεδομένων
- Χρησιμοποιεί, εσωτερικά, αντίστοιχες κλήσεις συστήματος `open`, `read`, `write`, `close`, ...


εφαρμογή (application)
fopen, fread,
fwrite, fclose, …
βιβλιοθήκη stdio
αποθήκη
user
open, read,
write, close, …
system
κώδικας συστήματος

Μορφοποιημένο σκανάρσιμα/εκτύπωση

- Μετατροπή ανάμεσα στην «εσωτερική» αναπαράσταση / κωδικοποίηση των δεδομένων που υπάρχουν στην μνήμη του προγράμματος, και μια (εντελώς) διαφορετική μορφή που είναι κατανοητή στον άνθρωπο (human readable)
- Η stdio παρέχει μια σειρά από τέτοιες λειτουργίες, για την ροή της καθιερωμένης εισόδου/εξόδου αλλά και για οποιαδήποτε άλλη ροή θελήσει να δημιουργήσει η εφαρμογή
  - `getc()`, `putc()`, `scanf()`, `printf()`, …
  - `fgetc()`, `fputc()`, `fscanf()`, `fprintf()`, …
- Η stdio δημιουργεί αυτόματως ανοιχτές ροές για την συμβατική είσοδο, έξοδο και έξοδο λαθών
  - `FILE *stdin, *stdout, *stderr`

Βασικές λειτουργίες αρχείων (1)

- `FILE *fopen(const char *filename, const char *mode)`
- Ο τρόπος με τον οποίο πρόκειται να χρησιμοποιηθεί το αρχείο προσδιορίζεται μέσω της παραμέτρου `mode`
  - `r`: για διάβασμα, το αρχείο πρέπει να υπάρχει
  - `w`: δημιουργία κενού αρχείου για γράψιμο, αν το αρχείο υπάρχει, σβήνονται όλα τα περιεχόμενα του και μένει κενό
  - `a`: για προσθήκη στο τέλος, το αρχείο δημιουργείται αν δεν υπάρχει
  - `r+`: για διάβασμα και γράψιμο, το αρχείο πρέπει να υπάρχει
  - `w+`: δημιουργία κενού αρχείου για διάβασμα και γράψιμο, αν το αρχείο υπάρχει, σβήνονται όλα τα περιεχόμενα του και μένει κενό
  - `a+`: για διάβασμα και προσθήκη στο τέλος, το αρχείο πρέπει να υπάρχει

Βασικές λειτουργίες αρχείων (2)

- `size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)`
  - ανάγνωση bytes με αντιγραφή στην διεύθυνση `ptr`
  - ο αριθμός των bytes που διαβάζονται είναι `size` x `nmemb`

- `size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)`
  - γραψιμο bytes με αντιγραφή από την διεύθυνση `ptr`
  - ο αριθμός των bytes που γράφονται είναι `size` x `nmemb`

- `int fclose(FILE *stream)`
  - κλείσιμο, με απελευθέρωση πόρων (εσωτερικών buffers)

Βασικές λειτουργίες αρχείων (3)

- `void rewind(FILE *stream)`
  - επαναφορά θέσης γράψιματος/ανάγνωσης στην αρχή
- `int fgetpos(FILE *stream, fpos_t *pos)`
  - επιστροφή θέσης γράψιματος/ανάγνωσης
- `int fsetpos(FILE *stream, const fpos_t *pos)`
  - απόλυτη μετακίνηση της θέσης γράψιματος/ανάγνωσης
- `int fseek(FILE *stream, long int offset, int whence)`
  - σχετική μετακίνηση της θέσης ανάγνωσης/εγγραφής
  - το `whence` ορίζει το σημείο αναφοράς για το `offset`
  - `SEEK_SET`: αρχή του αρχείου
  - `SEEK_CUR`: τρέχουσα θέση γραψίματος/ανάγνωσης
  - `SEEK_END`: τέλος του αρχείου

Βασικές λειτουργίες αρχείων (4)

- `int feof(FILE *stream)`
  - έλεγχος τέλους αρχείου
- `int fflush(FILE *stream)`
  - άδειασμα εσωτερικής αποθήκης
  - για αρχεία: γράψιμο στον δίσκο
- `int ferror(FILE *stream)`
  - έλεγχος για λάθος (από προηγούμενη λειτουργία)
- `void perror(const char *str)`
  - εκτύπωση μηνύματος λάθος στην συμβατική έξοδο λαθών (stderr)
- `void clearerr(FILE *stream)`
  - καθάρισμα (reset) λάθους και ένδειξης eof

Προσοχή στις λεπτομέρειες

- Διαβάστε **προσεκτικά** το manual
- Υπάρχουν διάφορα «περίεργα» ...
- Για παράδειγμα:
  - If a file is opened for update (the + mode), an output operation may not be followed by an input operation without flushing the buffer or repositioning, and an input operation may not be followed by an output operation without flushing the buffer or repositioning, unless the input operation has reached end-of-file.
  - For input streams, fflush() discards any buffered data that has been fetched from the underlying file, but has not been consumed by the application.
  - For output streams, fflush() forces a write of all user-space buffered data for the given output or update stream via the stream's underlying write function – for files, this does not ensure that the data will actually be written on disk.

```
FILE *f;
char str[]="hello world";

f=fopen("test","wb+");
fwrite(str,1,5,f); fflush(f);
fwrite(&str[5],1,6,f); fflush(f);
fwrite(" :-)",1,4,f); fflush(f);
…
```

str

h	e	l	l	o		w	o	r	l	d	\0
68	65	6C	6C	6F	20	77	6F	72	6C	64	00

← ερμηνεία ως ASCII

← τιμή bytes στην μνήμη

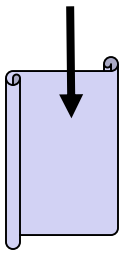
```
FILE *f;
char str[]="hello world";

f=fopen("test","wb+");
fwrite(str,1,5,f); fflush(f);
fwrite(&str[5],1,6,f); fflush(f);
fwrite(" :-)",1,4,f); fflush(f);
…
```

str

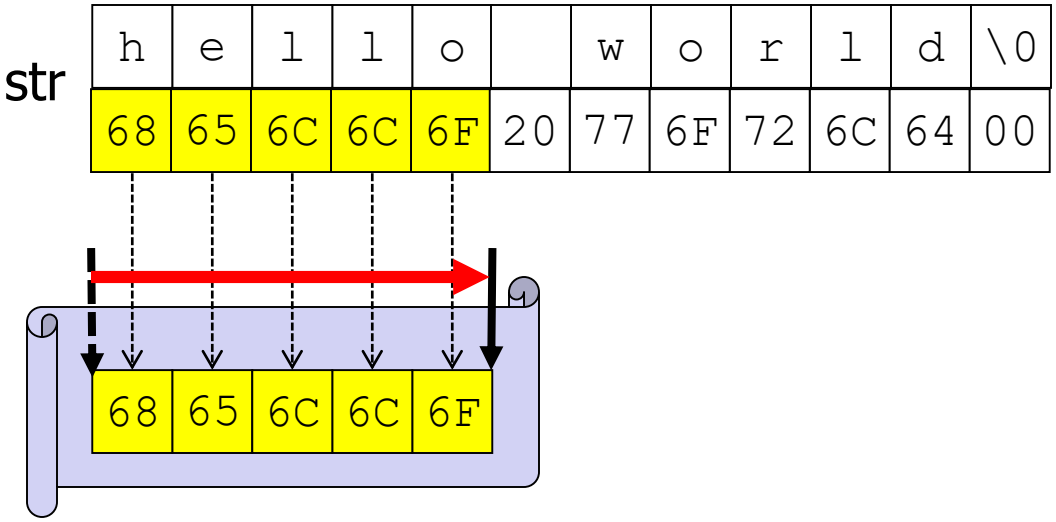
h	e	l	l	o		w	o	r	l	d	\0
68	65	6C	6C	6F	20	77	6F	72	6C	64	00

test



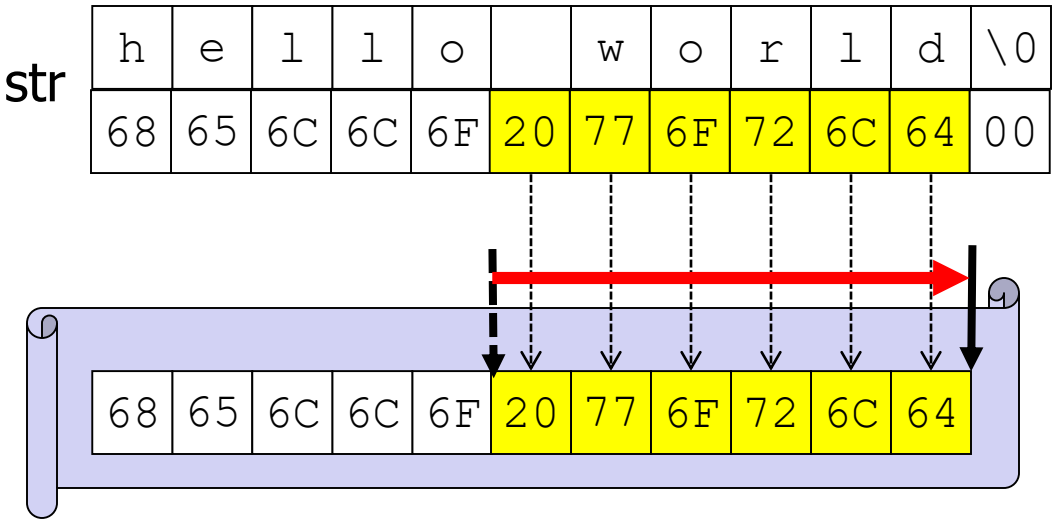
```
FILE *f;
char str[]="hello world";

f=fopen("test","wb+");
fwrite(str,1,5,f); fflush(f);
fwrite(&str[5],1,6,f); fflush(f);
fwrite(" :-)",1,4,f); fflush(f);
…
```



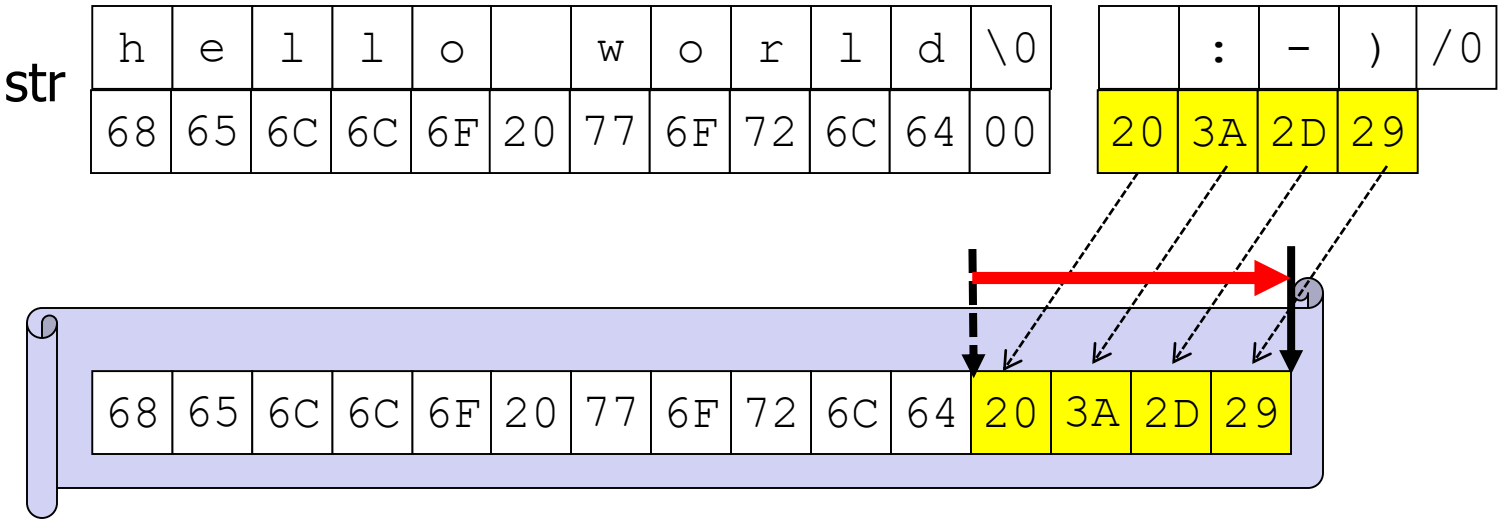
```
FILE *f;
char str[]="hello world";

f=fopen("test","wb+");
fwrite(str,1,5,f); fflush(f);
fwrite(&str[5],1,6,f); fflush(f);
fwrite(" :-)",1,4,f); fflush(f);
…
```



```
FILE *f;
char str[]="hello world";

f=fopen("test","wb+");
fwrite(str,1,5,f); fflush(f);
fwrite(&str[5],1,6,f); fflush(f);
fwrite(" :-)",1,4,f); fflush(f);
…
```



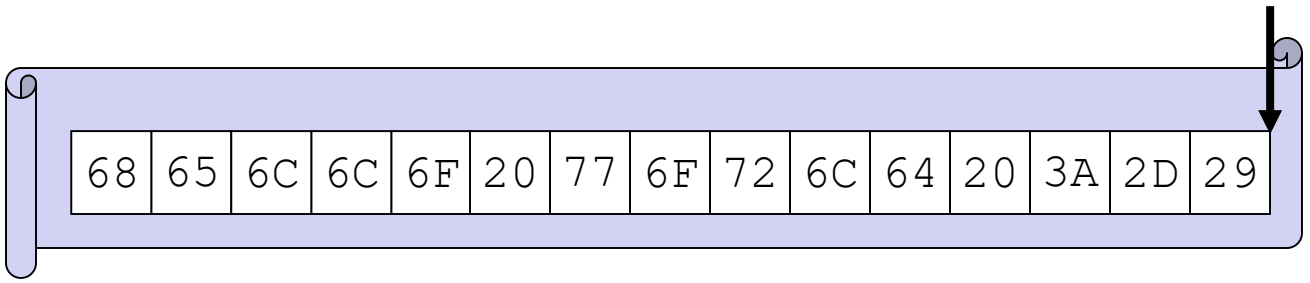
```
FILE *f;
char str[]="hello world";

f=fopen("test","wb+");
fwrite(str,1,5,f); fflush(f);
fwrite(&str[5],1,6,f); fflush(f);
fwrite(" :-)",1,4,f); fflush(f);
…
```

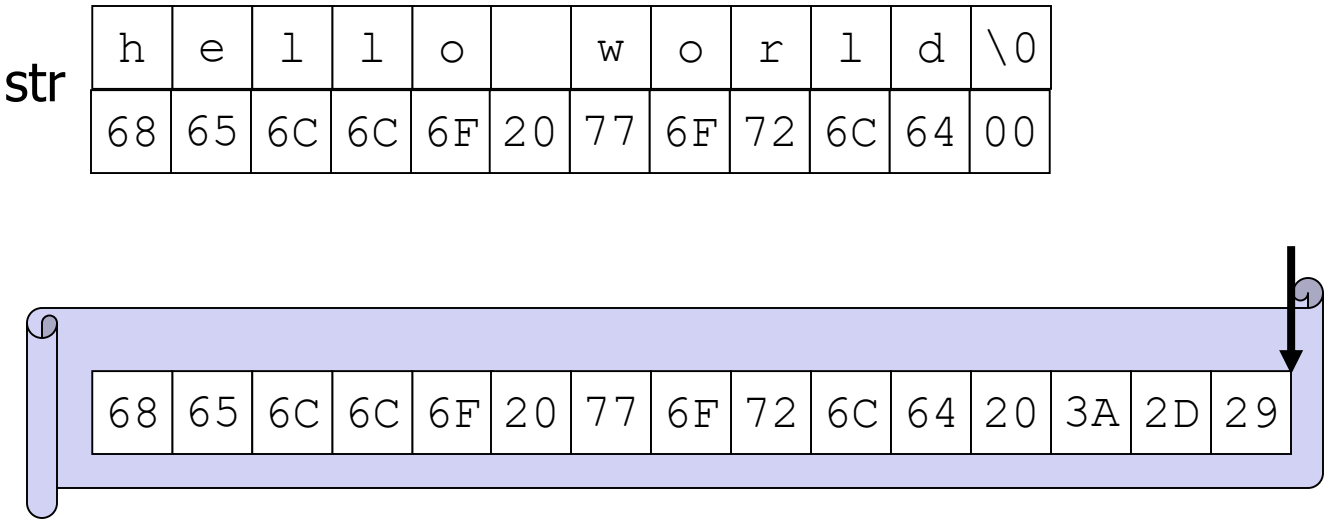
str

h	e	l	l	o		w	o	r	l	d	\0
68	65	6C	6C	6F	20	77	6F	72	6C	64	00

test



```
…
fseek(f,-3,SEEK_CUR);
fread(&str[6],1,3,f);
fseek(f,-9,SEEK_END);
fwrite("there",1,5,f); fflush(f);
fclose(f);
```

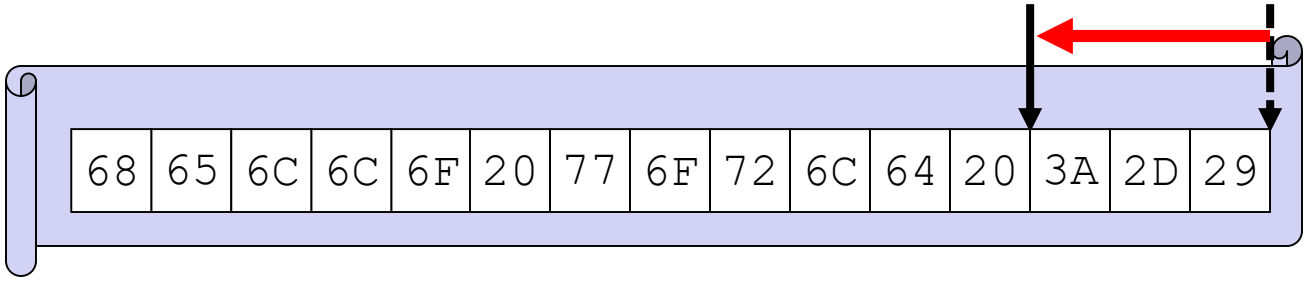


```
…
fseek(f,-3,SEEK_CUR);
fread(&str[6],1,3,f);
fseek(f,-9,SEEK_END);
fwrite("there",1,5,f); fflush(f);
fclose(f);
```

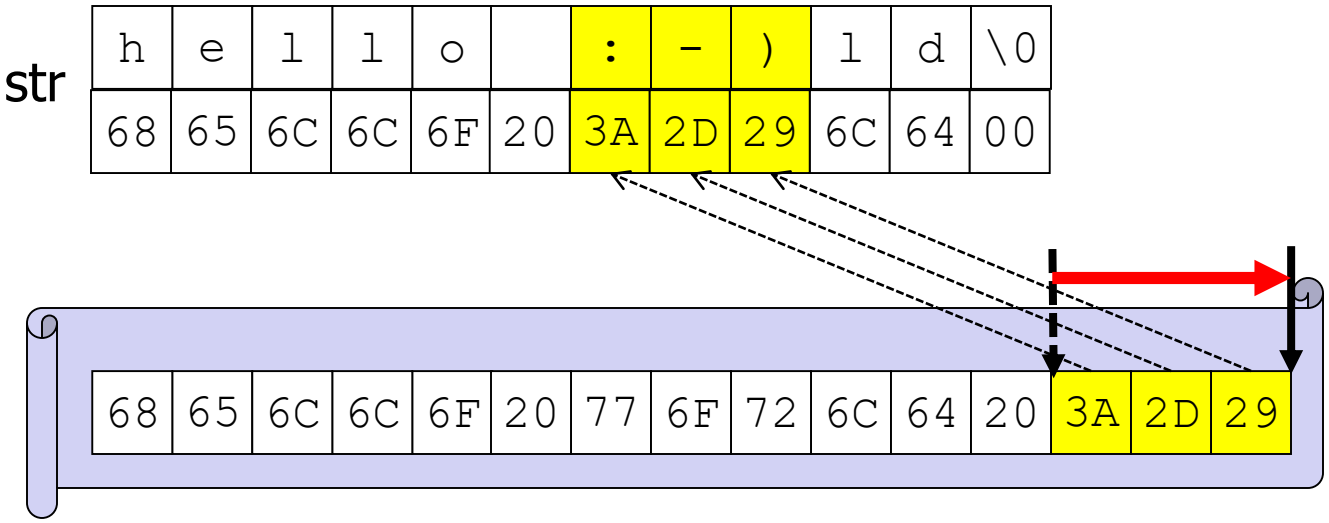
str

h	e	l	l	o		w	o	r	l	d	\0
68	65	6C	6C	6F	20	77	6F	72	6C	64	00

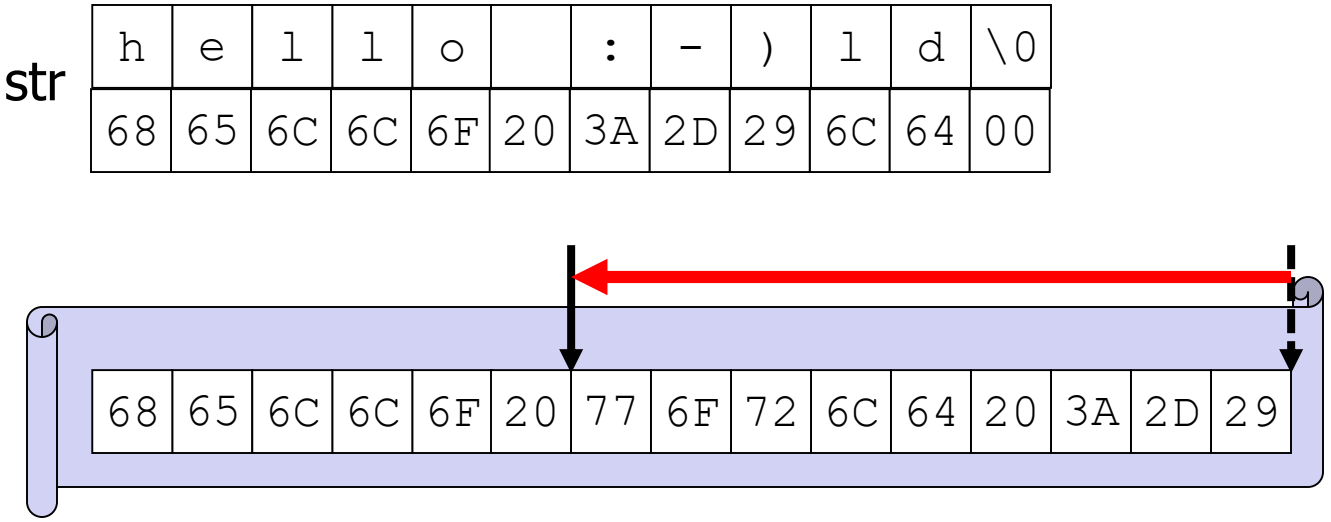
test



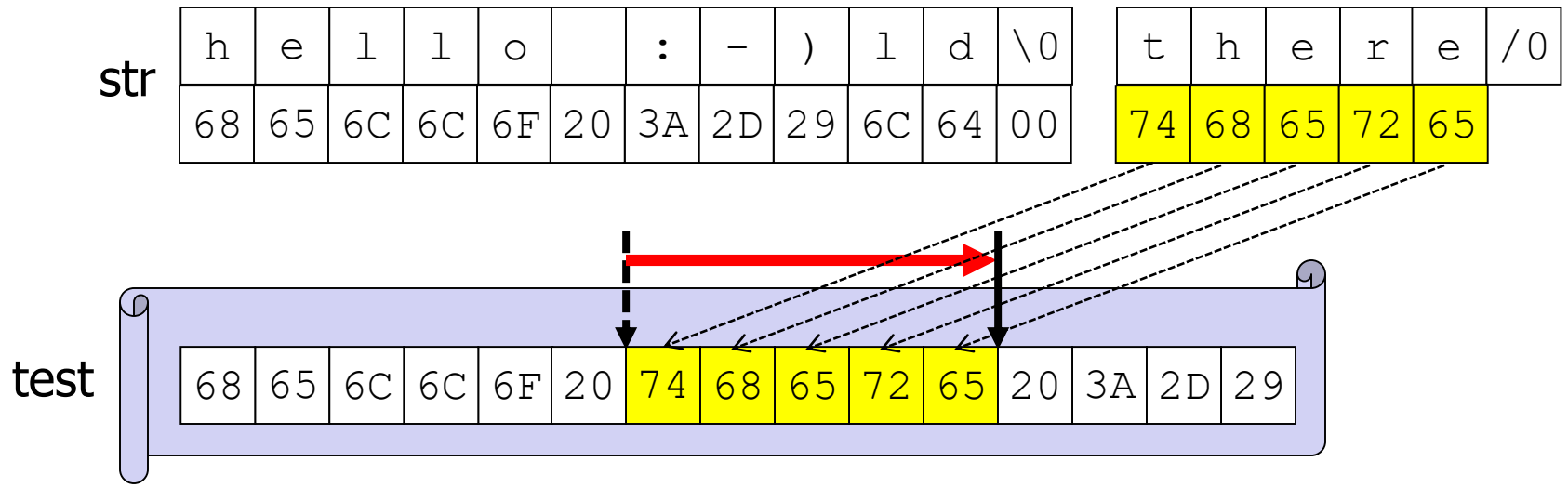
```
…
fseek(f,-3,SEEK_CUR);
fread(&str[6],1,3,f);
fseek(f,-9,SEEK_END);
fwrite("there",1,5,f); fflush(f);
fclose(f);
```



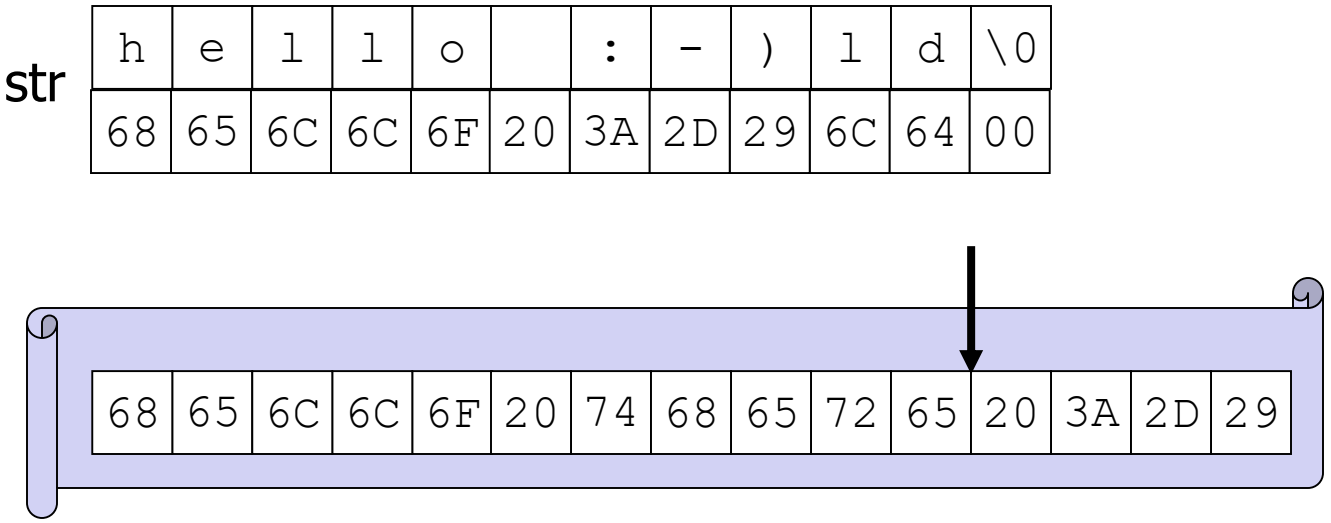
```
…
fseek(f,-3,SEEK_CUR);
fread(&str[6],1,3,f);
fseek(f,-9,SEEK_END);
fwrite("there",1,5,f); fflush(f);
fclose(f);
```



```
…
fseek(f,-3,SEEK_CUR);
fread(&str[6],1,3,f);
fseek(f,-9,SEEK_END);
fwrite("there",1,5,f); fflush(f);
fclose(f);
```



```
…
fseek(f,-3,SEEK_CUR);
fread(&str[6],1,3,f);
fseek(f,-9,SEEK_END);
fwrite("there",1,5,f); fflush(f);
fclose(f);
```



```
…
fseek(f,-3,SEEK_CUR);
fread(&str[6],1,3,f);
fseek(f,-9,SEEK_END);
fwrite("there",1,5,f); fflush(f);
fclose(f);
```

str

h	e	l	l	o		:	-	)	l	d	\0
68	65	6C	6C	6F	20	3A	2D	29	6C	64	00

test

68	65	6C	6C	6F	20	74	68	65	72	65	20	3A	2D	29

Κόψιμο/επέκταση αρχείου με `truncate`

```
int truncate(const char *path,
             off_t length);
```

- Αν η τιμή `length` είναι μικρότερη από το μήκος του αρχείου, το αρχείο **κόβεται** σε αυτό το μήκος
- Αν η τιμή `length` είναι μεγαλύτερη από το μήκος του αρχείου, το αρχείο **επεκτείνεται** σε αυτό το μήκος – τα επιπλέον bytes έχουν τιμή `x00`

- Λειτουργία του συστήματος (όχι της stdio)
  - καλό είναι να **αποφεύγεται** η χρήση λειτουργιών συστήματος παράλληλα με λειτουργίες της stdio

Μορφοποιημένο σκανάρισμα / εκτύπωση

- Αφορά ροές δεδομένων ASII (όχι binary)

- Η `scanf/fscanf` διαβάζει σύμβολα που **διαχωρίζονται** από έναν ή περισσότερους «κατά σύμβαση» διαχωριστικούς χαρακτήρες
  - αυτοί δεν επιστρέφονται στο πρόγραμμα

- Η `printf/fprintf` εκτυπώνει **όλους** τους χαρακτήρες που δίνει το πρόγραμμα
  - δεν αγνοούνται οι «λευκοί» χαρακτήρες που περιέχει η έκφραση (string) που δίνεται από το πρόγραμμα

Εσωτερική αποθήκη

- Η stdio διατηρεί τους χαρακτήρες που προορίζονται για γράψιμο ή ανάγνωση σε **εσωτερική αποθήκη**
  - μείωση των κλήσεων συστήματος (`write`/`read`)
- **Γράψιμο:** τα δεδομένα γράφονται στην αποθήκη και μεταφέρονται στην ροή όταν γεμίσει η αποθήκη ή το ζητήσει ρητά το πρόγραμμα – με `fflush()`
- **Διάβασμα:** τα δεδομένα διαβάζονται από την ροή στην αποθήκη, και από εκεί προωθούνται στο πρόγραμμα
- Ξεχωριστή εσωτερική αποθήκη για κάθε ροή, που δημιουργείται **αυτόματα** κατά την **πρώτη** κλήση μιας λειτουργίας ανάγνωσης/γραψίματος

Πολιτική εσωτερικής αποθήκευσης

- Υποστηρίζονται 3 διαφορετικές πολιτικές

- **Line buffered:** άδειασμα σε `\n`
- **Fully buffered:** άδειασμα όταν η αποθήκη γεμίσει
- **Not buffered:** χωρίς ενδιάμεση αποθήκευση

- Η stdio θέτει **αυτόματα** την πολιτική αποθήκευσης, ανάλογα με τον «τύπο» της ροής δεδομένων
- Καθιερωμένη είσοδος/έξοδος: line buffered
- Καθιερωμένη έξοδος λαθών: not buffered
- Αρχεία: fully buffered

Αλλαγή πολιτικής αποθήκευσης

- Ο προγραμματιστής μπορεί να **προσδιορίσει** την επιθυμητή πολιτική αποθήκευσης, για κάθε ροή (`FILE`)

- ```
  int setvbuf(FILE *stream, char *buffer,
              int mode, size_t size);
  ```

- Καθορίζει την πολιτική αποθήκευσης, το μέγεθος της αποθήκης ή/και την μνήμη που θα χρησιμοποιηθεί για την αποθήκη

- Η λειτουργία `setvbuf` πρέπει να κληθεί **μετά** την `fopen` **προτού** κληθεί μια λειτουργία ανάγνωσης ή γραψίματος
  ```

```
int main(int argc , char *argv[]) {
  char str[]="hello\nworld\n";
  int i;

  if (argc > 1) {
    if (argv[1][0] == 'n') {
      setvbuf(stdout,NULL,_IONBF,0);
    }
    else if (argv[1][0] == 'f') {
      setvbuf(stdout,NULL,_IOFBF,0);
    }
  }
  printf("start printing\n");
  for (i=0; i<strlen(str); i++) {
    sleep(1);
    printf("%c",str[i]);
  }
  return(0);
}
```

not buffered

fully buffered

Ανάγνωση δυαδικών δεδομένων (binary)

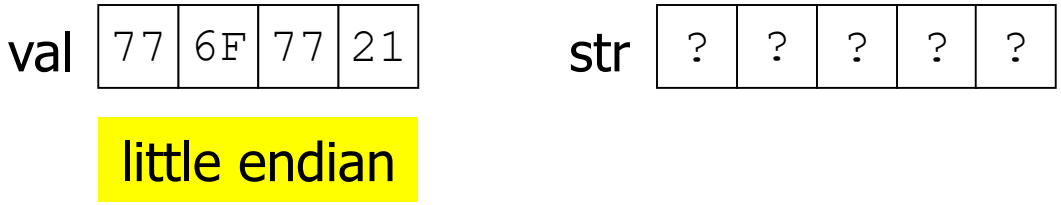
- Μια ροή μπορεί να περιέχει δεδομένα σε δυαδική μορφή
- Κλασικό παράδειγμα: binary file

- Το πρόγραμμα (ο προγραμματιστής) έχει την ευθύνη για την κατάλληλη / «σωστή» **ερμηνεία** των δεδομένων

- Όπως πάντα: τα ίδια δεδομένα μπορεί να ερμηνευθούν με εντελώς **διαφορετικούς** τρόπους ...

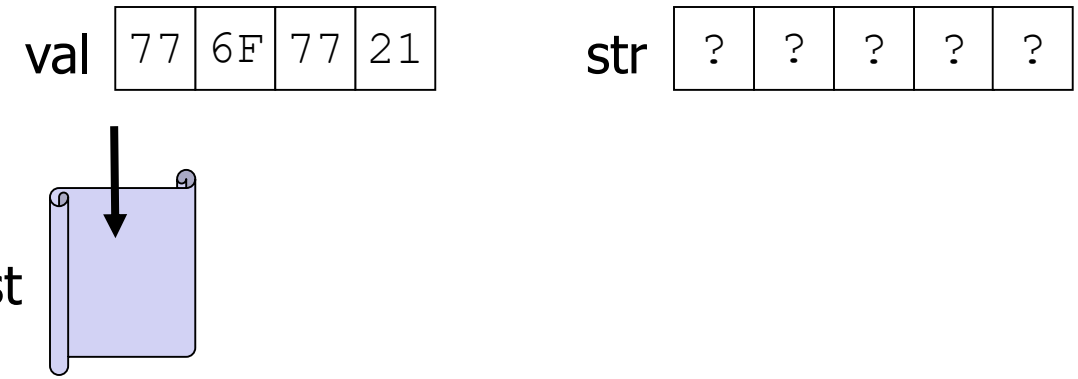
```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```



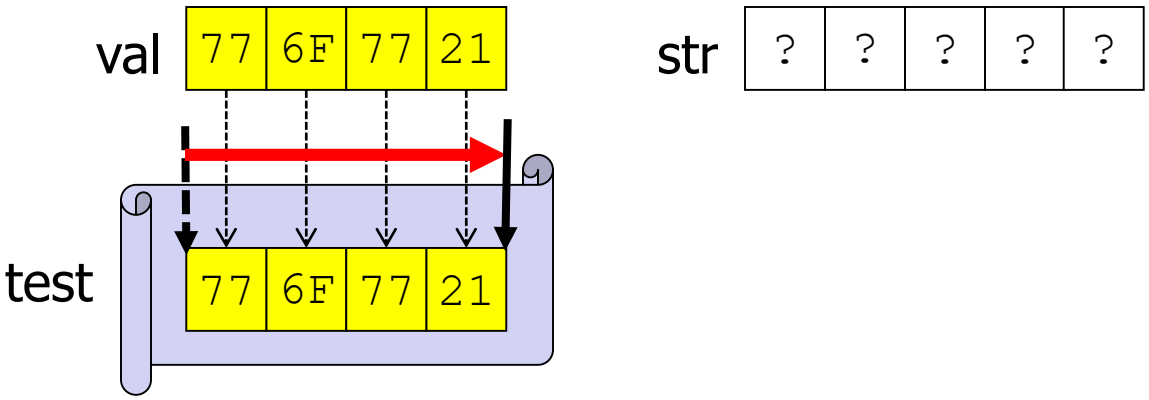
```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```



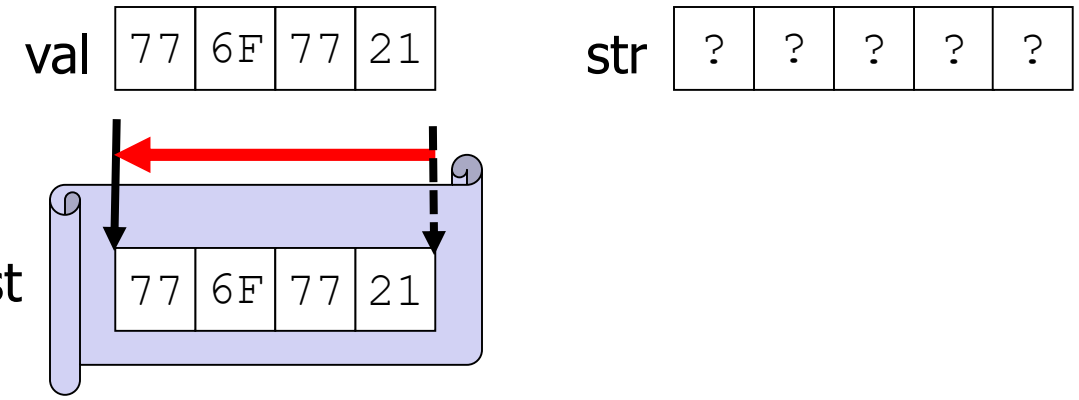
```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```



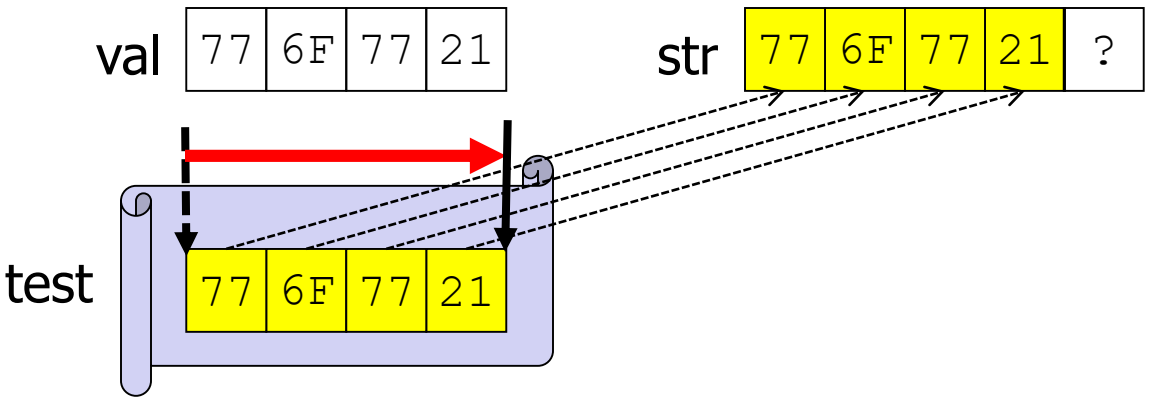
```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```



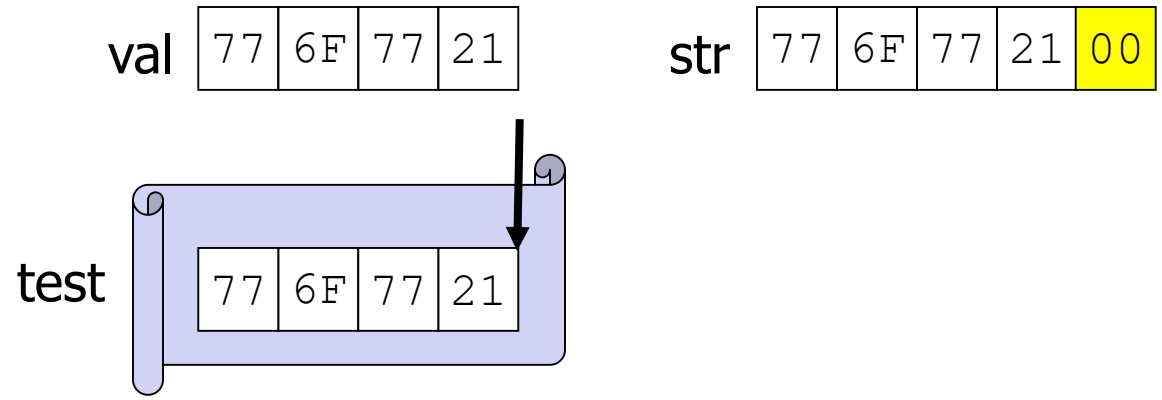
```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```



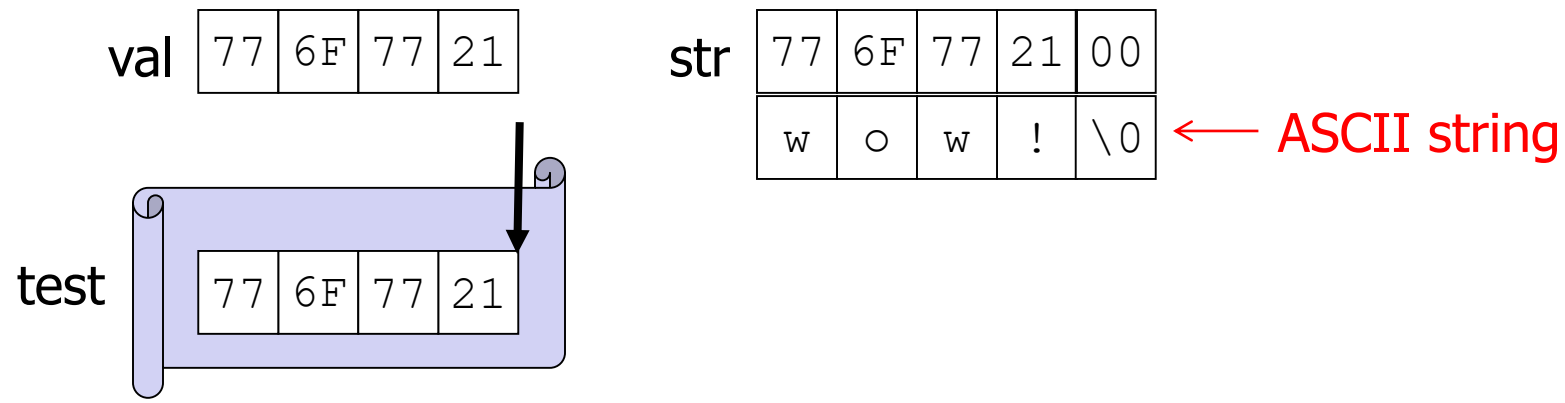
```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```



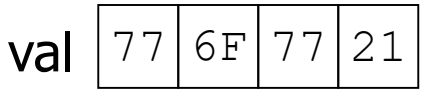
```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

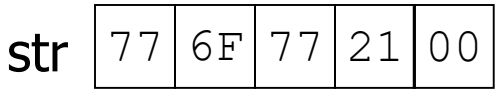
f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```

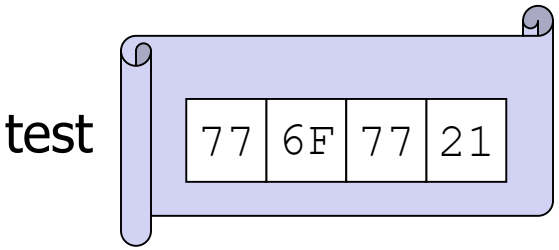


```
FILE *f;
int val=0x21776F77;
char str[sizeof(int)+1];

f = fopen("test","wb+");
fwrite(&val,sizeof(int),1,f); fflush(f);
printf("wrote %x\n",val);
rewind(f);
fread(str,sizeof(char),sizeof(int),f);
str[sizeof(int)]='\0';
printf("read %s\n",str);
fclose(f);
```







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

- Οι καταλήξεις αρχείων (π.χ., `.pdf`) είναι απλά μια **σύμβαση** για πιο εύκολη αναζήτηση από τον χρήστη
  - σε πιο ψηλό επίπεδο από ίδιο το σύστημα αρχείων
  - καθαρά ενδεικτική για τα περιεχόμενα του αρχείου (ένα αρχείο με κατάληξη .pdf δεν σημαίνει ότι *είναι* pdf)