Προγραμματισμός ΙΙ (ECE116)

#13 λειτουργίες αρχείων σε επίπεδο συστήματος

Σπύρος Λάλης
lalis@uth.gr

Βασικές λειτουργίες (1)

- `int open(const char *path,`
  `int flags, mode_t perms)`
  - ανοίγει το αρχείο με όνομα `path` (αν συμπεριλαμβάνεται ένα μονοπάτι καταλόγων, αυτό πρέπει ήδη να υφίσταται)
- Η επιθυμητή πρόσβαση προσδιορίζεται μέσω `flags`
  - `O_RDONLY`,`O_WRONLY`,`O_RDWR`: διάβασμα, γράψιμο, και τα δύο
  - `O_APPEND`,`O_TRUNC`: προσθήκη στο τέλος, κόψιμο αρχείου
  - `O_CREAT`: δημιουργία αρχείου, αν δεν υπάρχει ήδη
  - `O_EXCL`: μαζί με `O_CREAT`, αποτυχία αν το αρχείο υπάρχει
- Οι άδειες πρόσβασης προσδιορίζονται μέσω `perms`
  - οκταδική τιμή (π.χ. `0700`) ή συμβολική σταθερά (π.χ. `S_IRWXU`)
  - λαμβάνονται υπόψη μόνο όταν δημιουργείται ένα **νέο** αρχείο

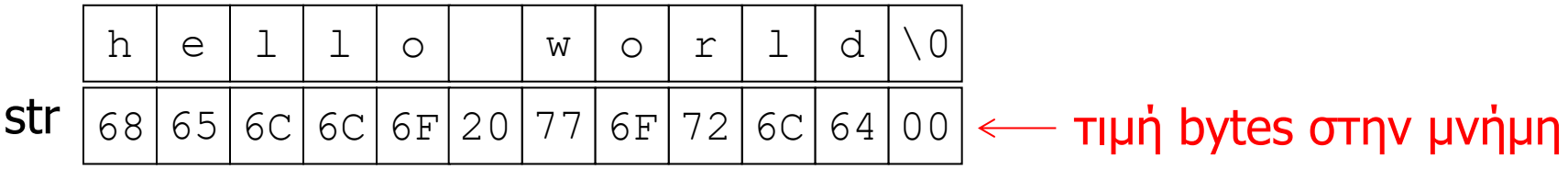
Περιγραφέας αρχείου (file descriptor)

- Η `open` επιστρέφει έναν **περιγραφέα αρχείου**
  - file descriptor
- Είναι ένας «απλός» **ακέραιος**
  - **δεν** είναι κάποιος pointer-to-struct όπως ίσως θα περίμενε κανείς …
- Χρησιμοποιείται ως **αναφορά** (reference) στο αρχείο
  - όλες οι υπόλοιπες λειτουργίες αρχείων δέχονται ως παράμετρο έναν περιγραφέα αρχείου (ακέραιο) **χωρίς** να αλλάζουν την τιμή του
- Για κάθε περιγραφέα, το λειτουργικό κρατά την αντίστοιχη θέση ανάγνωσης/εγγραφής στο αρχείο
  - η θέση ανάγνωσης/εγγραφής **δεν** είναι άμεσα προσπελάσιμη
  - ο περιγραφέας **δεν** «είναι» η θέση ανάγνωσης/εγγραφής
- Περισσότερα για τους περιγραφείς αρχείων, αργότερα

Βασικές λειτουργίες (2)

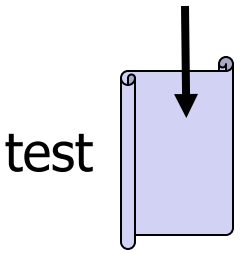
- `ssize_t read(int fd, void *buf, size_t n)`
  - ανάγνωση δεδομένων από το αρχείο (με μετακίνηση της θέσης ανάγνωσης/εγγραφής του περιγραφέα)
- `ssize_t write(int fd, void *buf, size_t n)`
  - εγγραφή δεδομένων στο αρχείο (με μετακίνηση της θέσης ανάγνωσης/εγγραφής του περιγραφέα)
- `off_t lseek(int fd, off_t pos, int whence)`
  - μετακίνηση θέσης ανάγνωσης/εγγραφής του περιγραφέα
- `int ftruncate(int fd, off_t length)`
  - κόψιμο/επέκταση του αρχείου
- `int fsync(int fd)`
  - σύγχρονη αποθήκευση δεδομένων στον δίσκο
- `int close(int fd)`
  - κλείσιμο του περιγραφέα

```
int fd;
char str[]="hello world";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,5);
write(fd,&str[5],6);
write(fd," :-)",4);
…
```

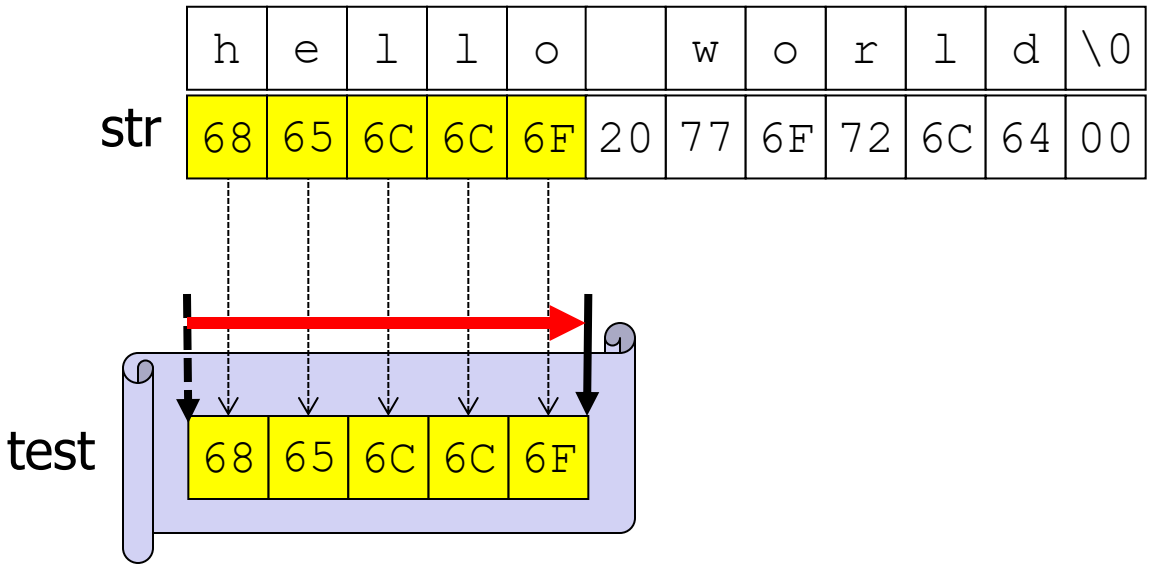


```
int fd;
char str[]="hello world";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,5);
write(fd,&str[5],6);
write(fd," :-)",4);
…
```

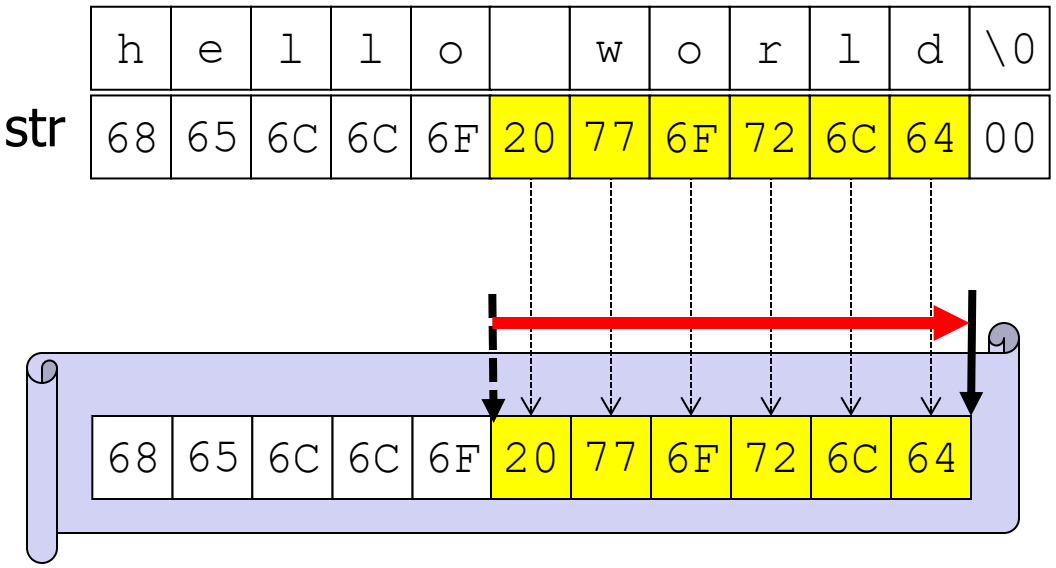
	h	e	l	l	o		w	o	r	l	d	\0
str	68	65	6C	6C	6F	20	77	6F	72	6C	64	00



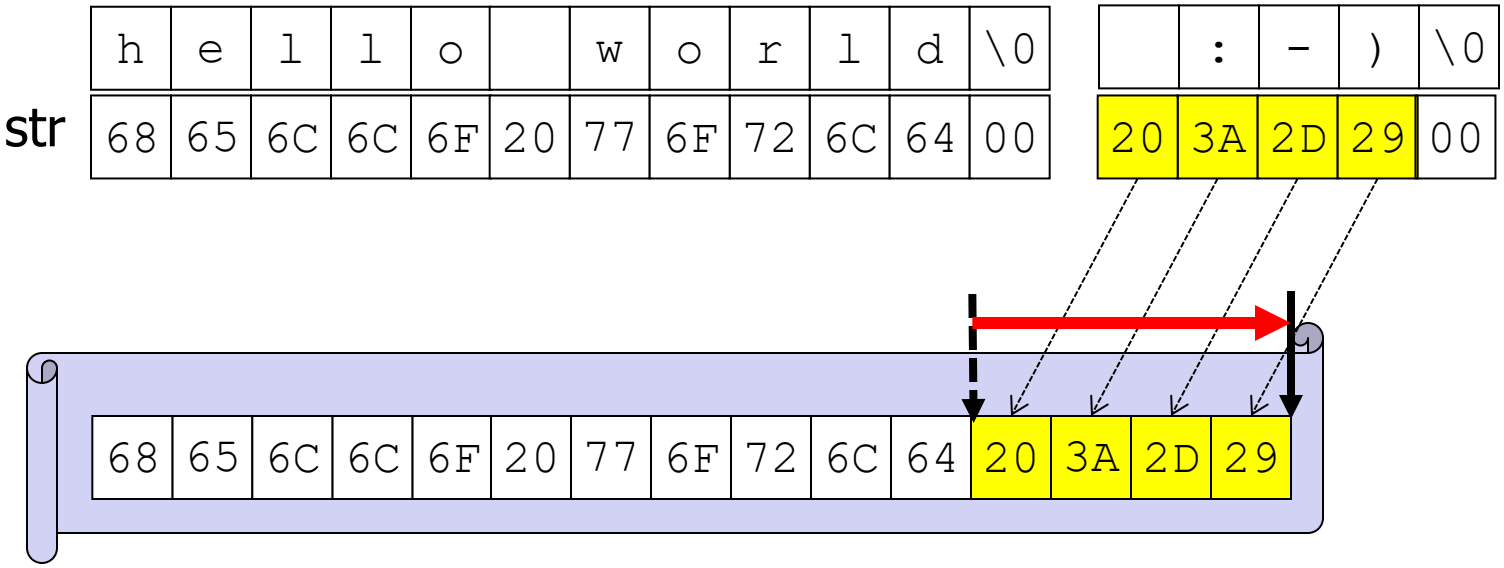
```
int fd;
char str[]="hello world";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,5);
write(fd,&str[5],6);
write(fd," :-)",4);
…
```



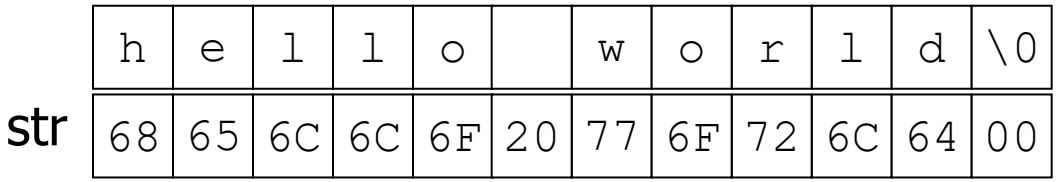
```
int fd;
char str[]="hello world";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,5);
write(fd,&str[5],6);
write(fd," :-)",4);
…
```

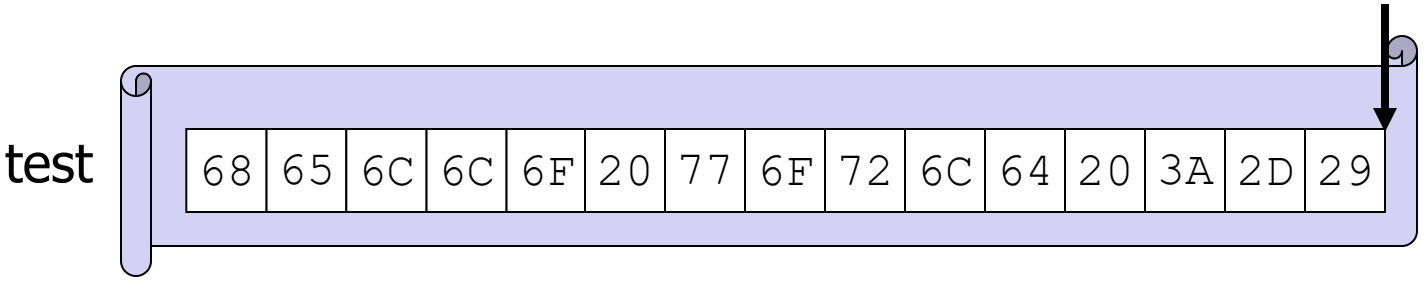


```
int fd;
char str[]="hello world";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,5);
write(fd,&str[5],6);
write(fd," :-)",4);
…
```

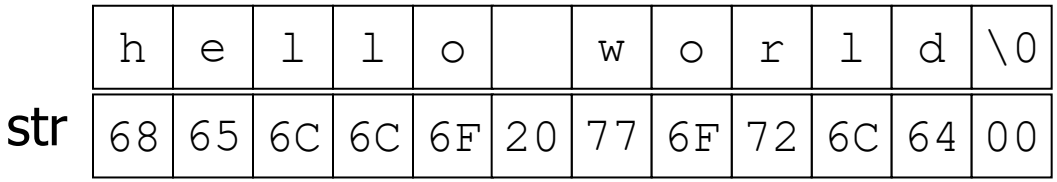


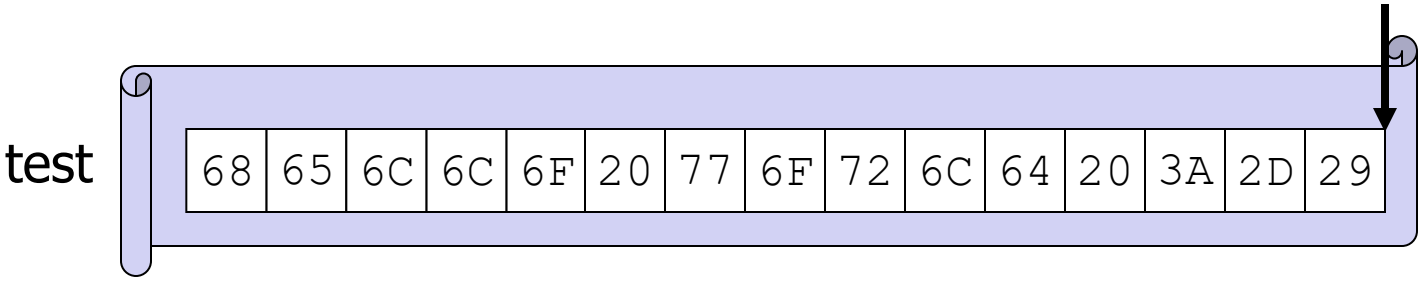
```
int fd;
char str[]="hello world";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,5);
write(fd,&str[5],6);
write(fd," :-)",4);
…
```



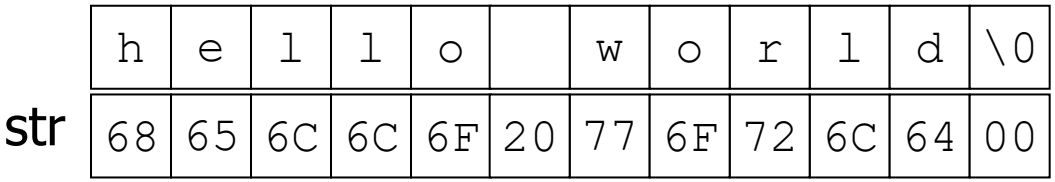


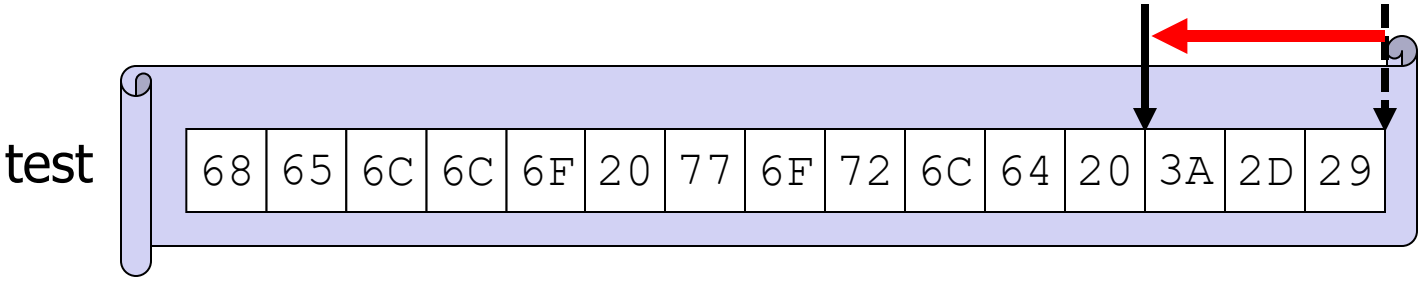
```
…
lseek(fd,(off_t)-3,SEEK_CUR);
read(fd,str,5);
read(fd,str,5);
lseek(fd,(off_t)-9,SEEK_CUR);
write(fd,"there",5);
…
```



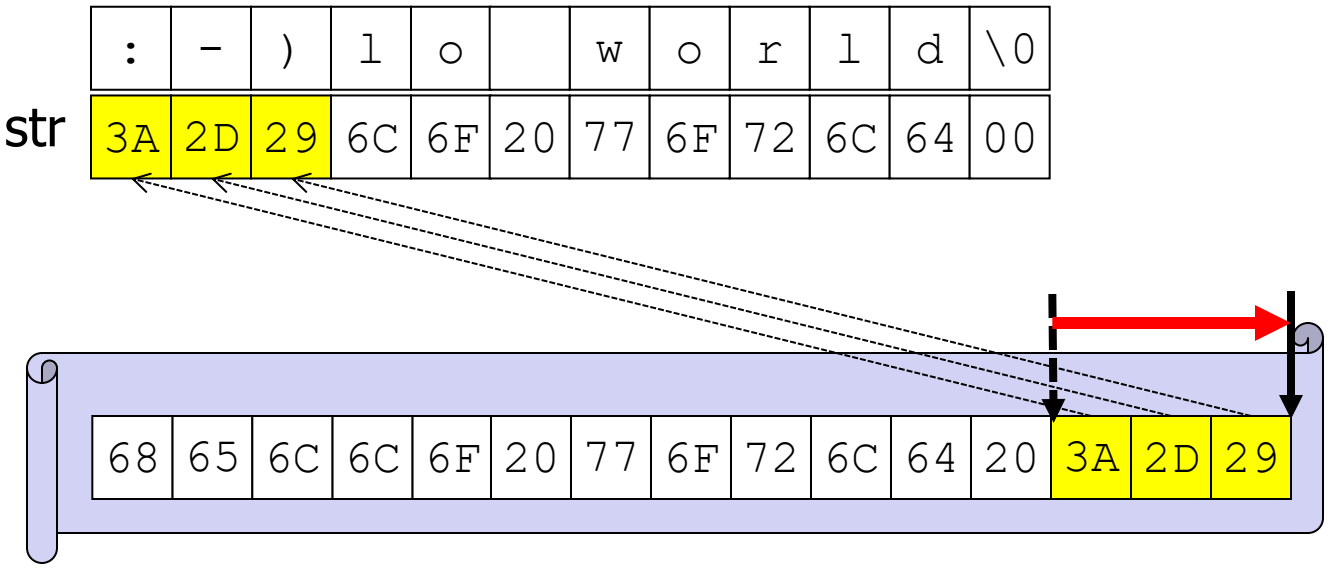


```
…
lseek(fd,(off_t)-3,SEEK_CUR);
read(fd,str,5);
read(fd,str,5);
lseek(fd,(off_t)-9,SEEK_CUR);
write(fd,"there",5);
…
```

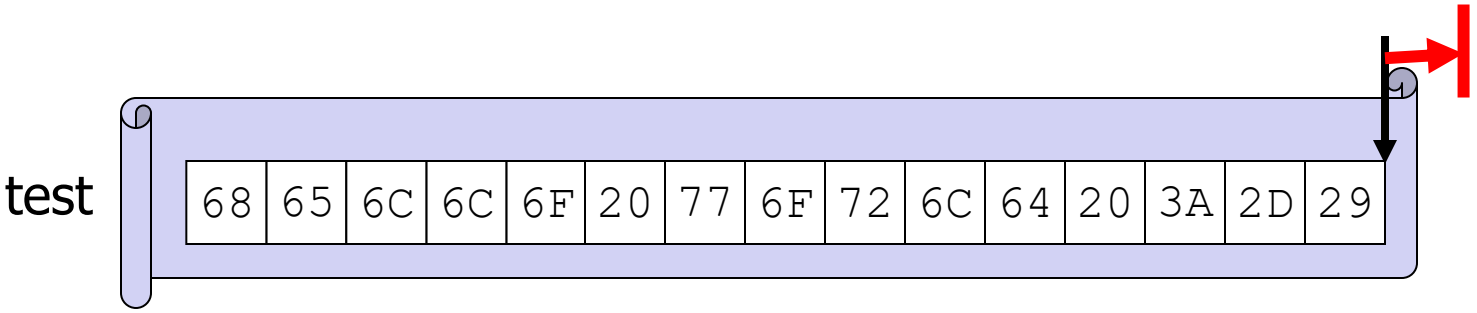




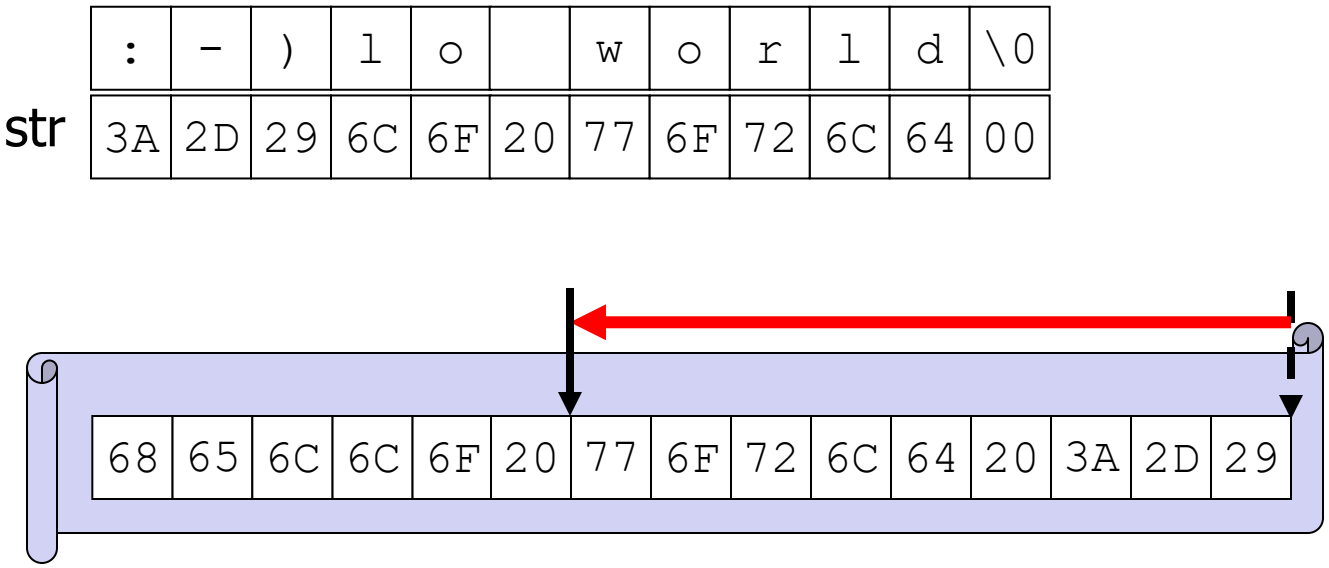
```
…
lseek(fd,(off_t)-3,SEEK_CUR);
read(fd,str,5);
read(fd,str,5);
lseek(fd,(off_t)-9,SEEK_CUR);
write(fd,"there",5);
…
```



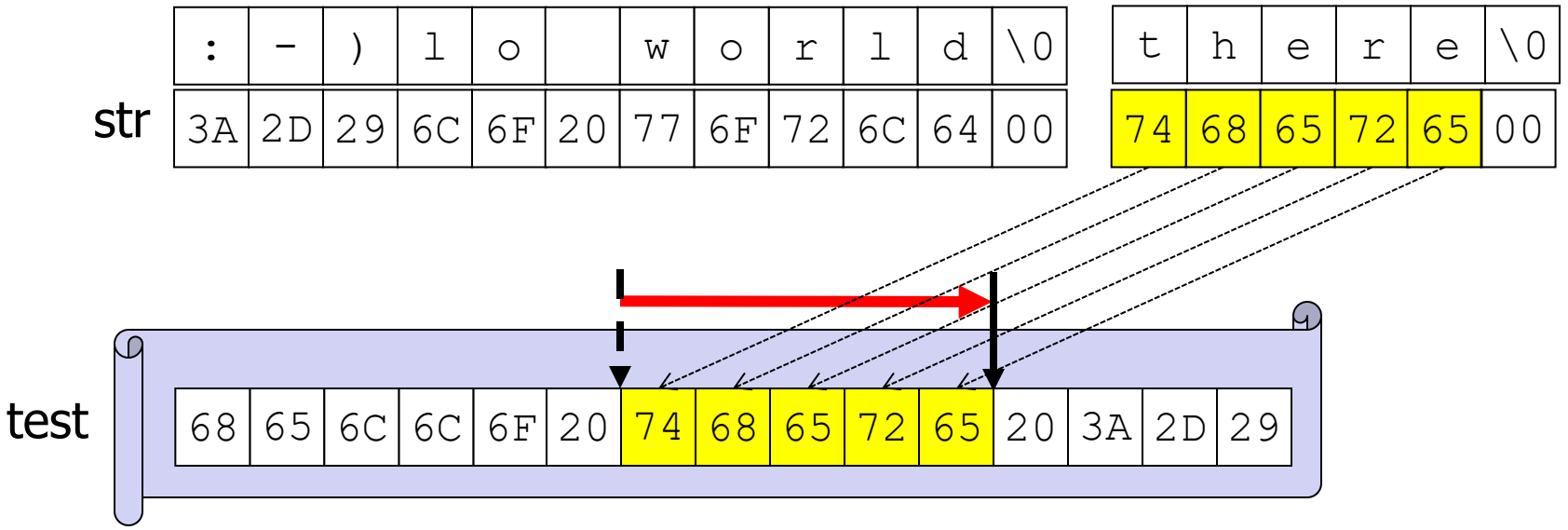
```
…
lseek(fd,(off_t)-3,SEEK_CUR);
read(fd,str,5);
read(fd,str,5);
lseek(fd,(off_t)-9,SEEK_CUR);
write(fd,"there",5);
…
```



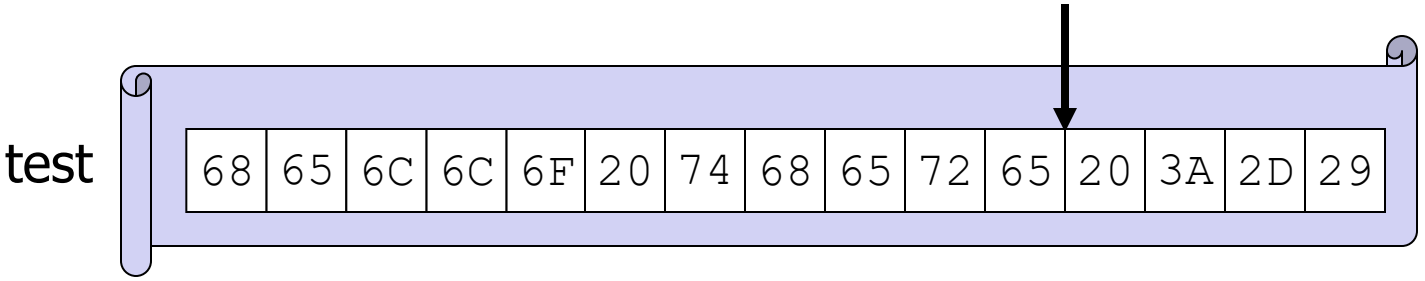
```
…
lseek(fd,(off_t)-3,SEEK_CUR);
read(fd,str,5);
read(fd,str,5);
lseek(fd,(off_t)-9,SEEK_CUR);
write(fd,"there",5);
…
```



```
…
lseek(fd,(off_t)-3,SEEK_CUR);
read(fd,str,5);
read(fd,str,5);
lseek(fd,(off_t)-9,SEEK_CUR);
write(fd,"there",5);
…
```

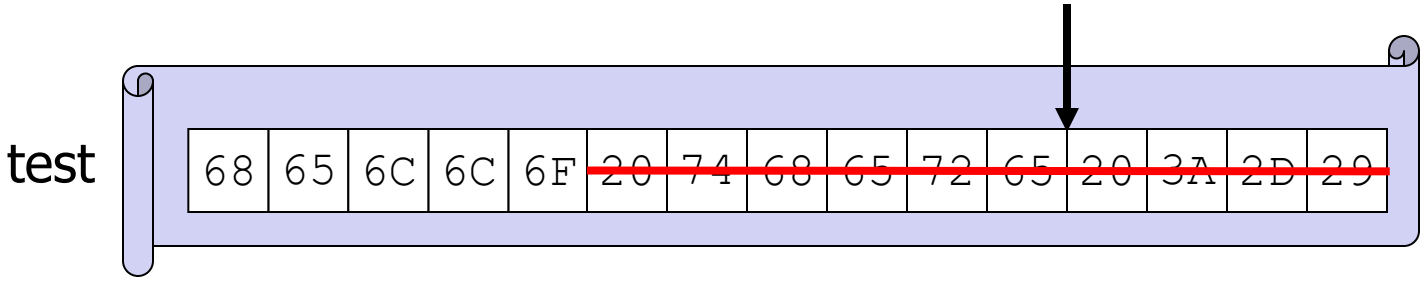


```
…
ftruncate(fd,5);
write(fd,&str[6],5);
close(fd);
```

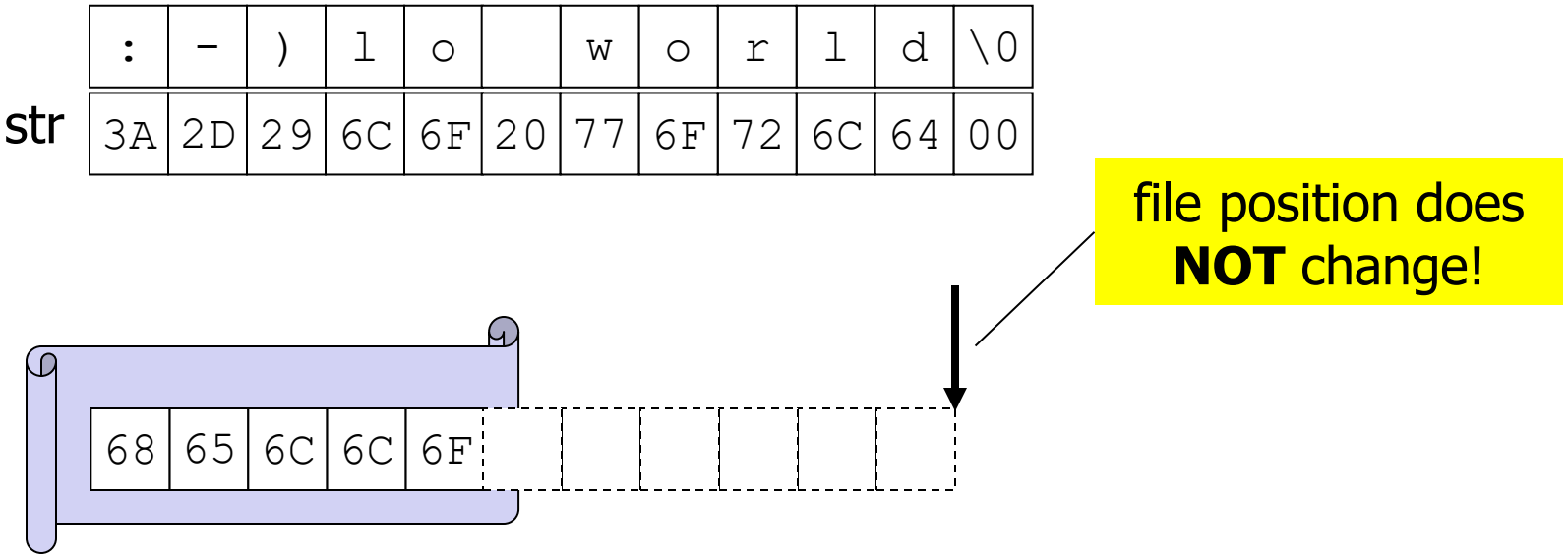


```
…
ftruncate(fd,5);
write(fd,&str[6],5);
close(fd);
```

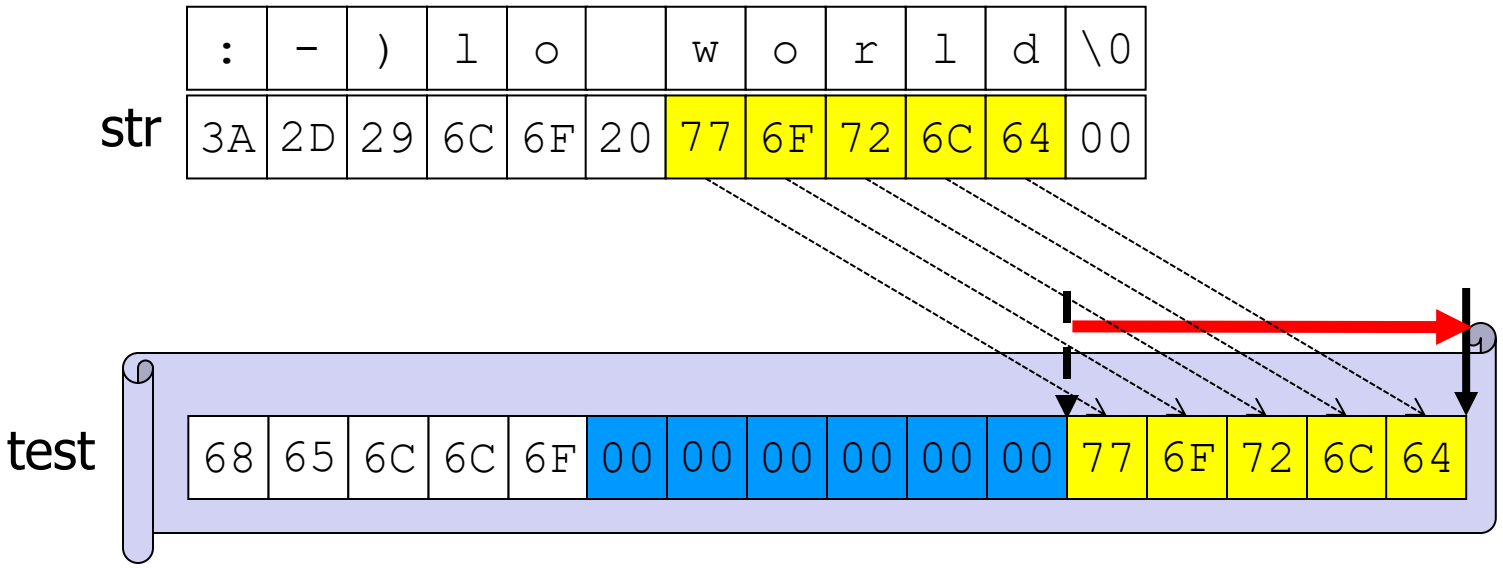
	:	-	)	l	o		w	o	r	l	d	\0
str	3A	2D	29	6C	6F	20	77	6F	72	6C	64	00



```
…
ftruncate(fd,5);
write(fd,&str[6],5);
close(fd);
```

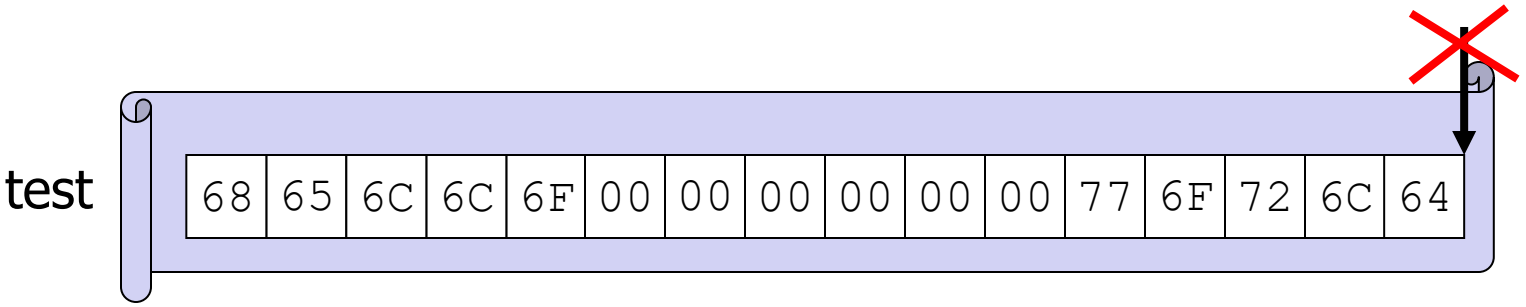


```
…
ftruncate(fd,5);
write(fd,&str[6],5);
close(fd);
```

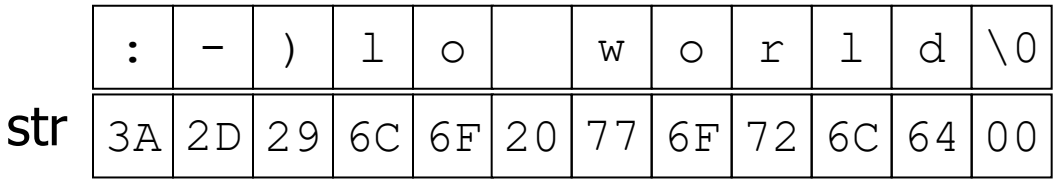


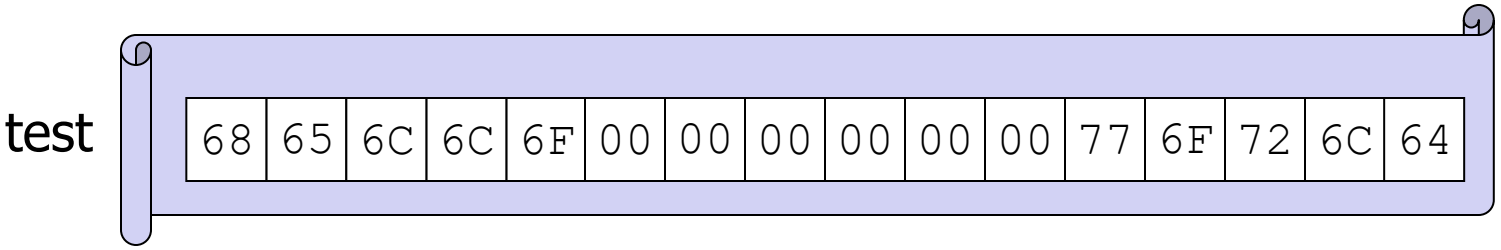
```
…
ftruncate(fd,5);
write(fd,&str[6],5);
close(fd);
```

	:	-	)	l	o		w	o	r	l	d	\0
str	3A	2D	29	6C	6F	20	77	6F	72	6C	64	00



```
…
ftruncate(fd,5);
write(fd,&str[6],5);
close(fd);
```





Προσοχή!

- Οι λειτουργίες **δεν** αλλάζουν τον περιγραφέα αρχείου
  - το ΛΣ αλλάζει (εσωτερικά) την θέση ανάγνωσης/εγγραφής που είναι **συσχετισμένη** με τον περιγραφέα
- Ξέρουμε ότι φτάσαμε στο τέλος του αρχείου **μόνο** αν η `read` διαβάσει `0` bytes (επιστρέψει τιμή 0)
- Οι `read`/`write` μπορεί να **μην** διαβάσουν/γράψουν όσα bytes ζήτησε ο προγραμματιστής
  - πρέπει να **ελέγχεται** η τιμή επιστροφής, και να **επαναληφθεί** η κλήση για να διαβαστούν/γραφτούν τα υπόλοιπα bytes
- Οι `write`/`close` **δεν** δίνουν εγγυήσεις σχετικά με την αποθήκευση των δεδομένων στον δίσκο
  - το λειτουργικό αποθηκεύει στο δίσκο **ασύγχρονα** – γιατί;
- Η **σύγχρονη** αποθήκευση στο δίσκο: με `fsync`

Προσοχή!

- Με την `lseek` η θέση ανάγνωσης/εγγραφής μπορεί να μετακινηθεί **πέρα** από το τέλος του αρχείου
  - αυτό **επιτρέπεται** – δεν επιστρέφεται κωδικός αποτυχίας!
- Αν από εκείνο το σημείο γραφτούν δεδομένα, το υπόλοιπο περιεχόμενο του αρχείου **γεμίζει** με `0`
  - αυτό γίνεται **αυτόματα** από το λειτουργικό

- Η `ftruncate` **δεν** αλλάζει την τρέχουσα θέση ανάγνωσης εγγραφής
  - μετά το κόψιμο, μπορεί να δείχνει πέρα από το τέλος του αρχείου
  - αν από εκείνο το σημείο γραφτούν δεδομένα, βλέπε παραπάνω
- Η `ftruncate` μπορεί να **επεκτείνει** το αρχείο
  - το επιπλέον περιεχόμενο του αρχείου γεμίζει με `0`

Μέγεθος αρχείου

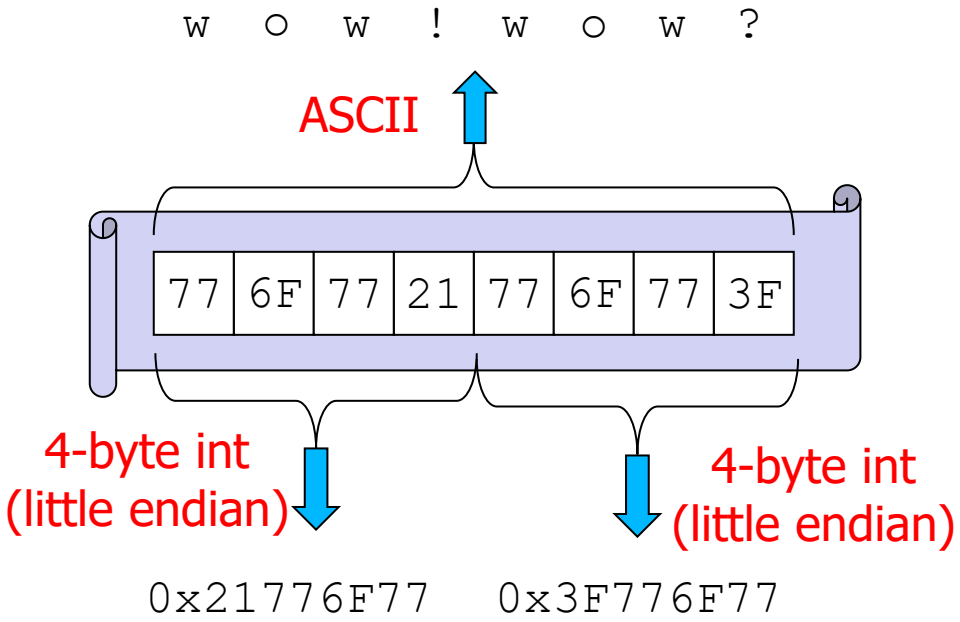
- Το πρόγραμμα μπορεί να μάθει το μέγεθος ενός αρχείου με δύο διαφορετικούς τρόπους

1. Μέσω της `lseek`, μετακινώντας την θέση ανάγνωσης/εγγραφής στο τέλος του αρχείου και λήψη της τιμής που επιστρέφεται
   - `flen = lseek(fd, 0, SEEK_END);`

2. Μέσω της λειτουργίας `stat/fstat` που επιστρέφει τα μεταδεδομένα του αρχείου (μεταξύ των οποίων και το τρέχον μέγεθος) μέσα σε μια δομή
   - περισσότερα για αυτό σε λίγο …

Δυαδικά δεδομένα (binary data)

- Τα δεδομένα γράφονται στα αρχεία και διαβάζονται από τα αρχεία σε **δυαδική** μορφή
  - όπως άλλωστε και τα δεδομένα στη μνήμη του Η/Υ
- Η ερμηνεία των δεδομένων είναι άλλη υπόθεση
  - τα ίδια δεδομένα μπορεί να ερμηνευτούν διαφορετικά



```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```

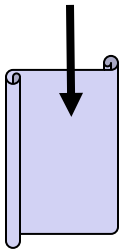
```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```

str	77	6F	77	21	77	6F	77	3F	00

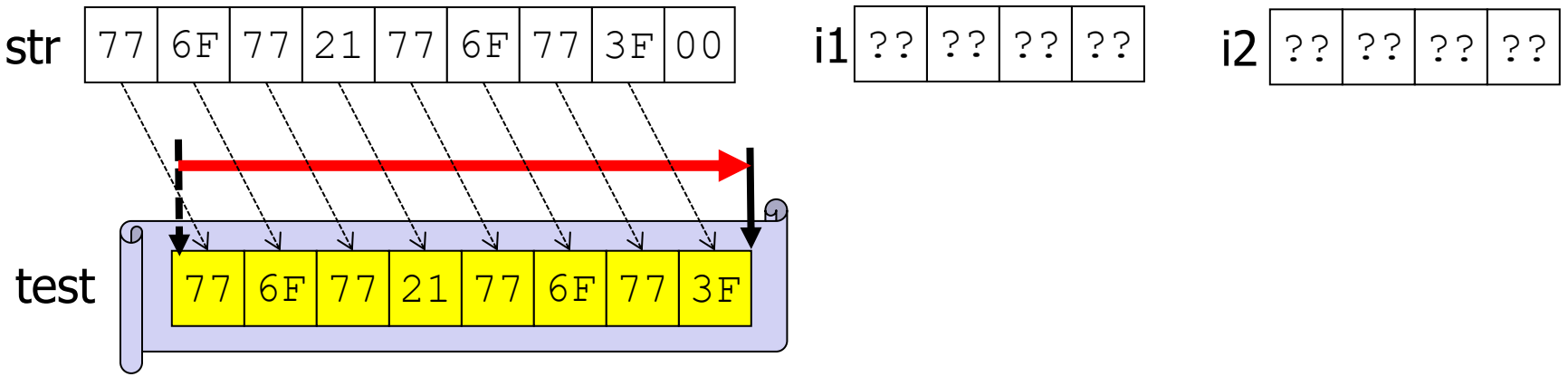
i1	??	??	??	??

i2	??	??	??	??

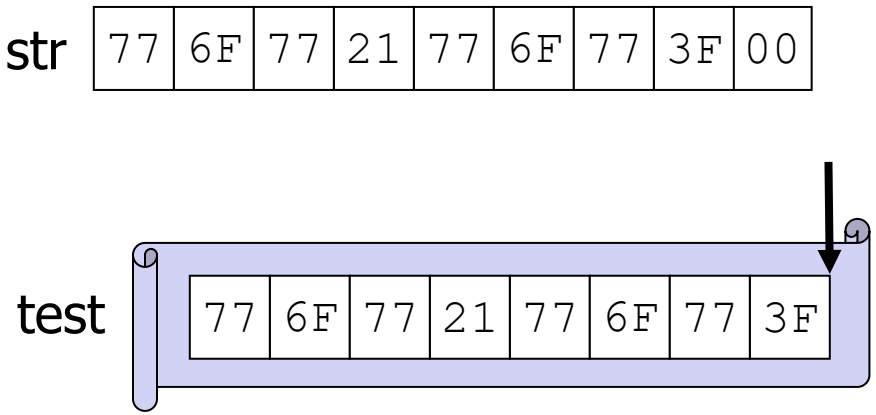
test



```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```



```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```

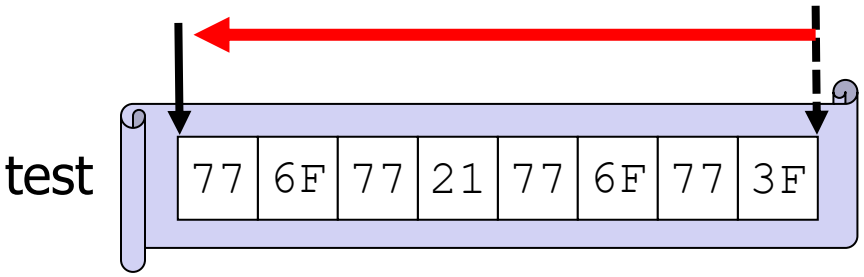


```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```

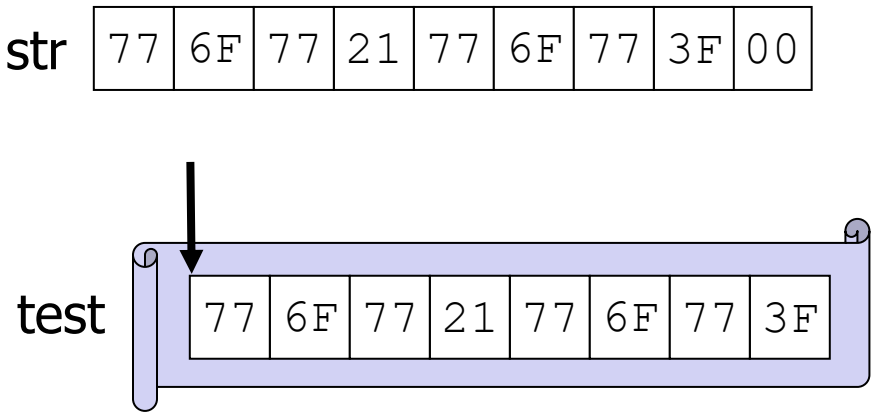
str	77	6F	77	21	77	6F	77	3F	00

i1	??	??	??	??

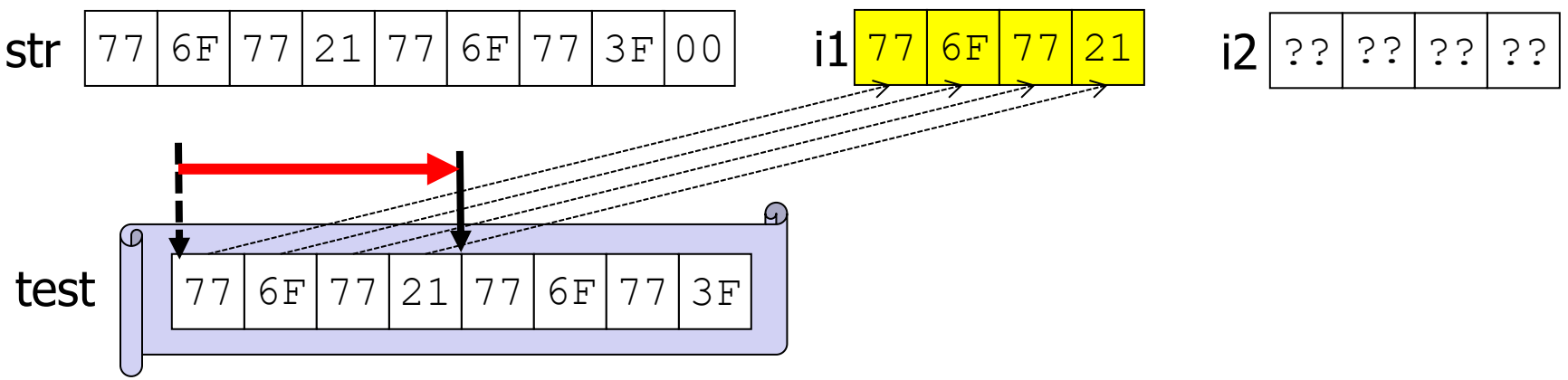
i2	??	??	??	??



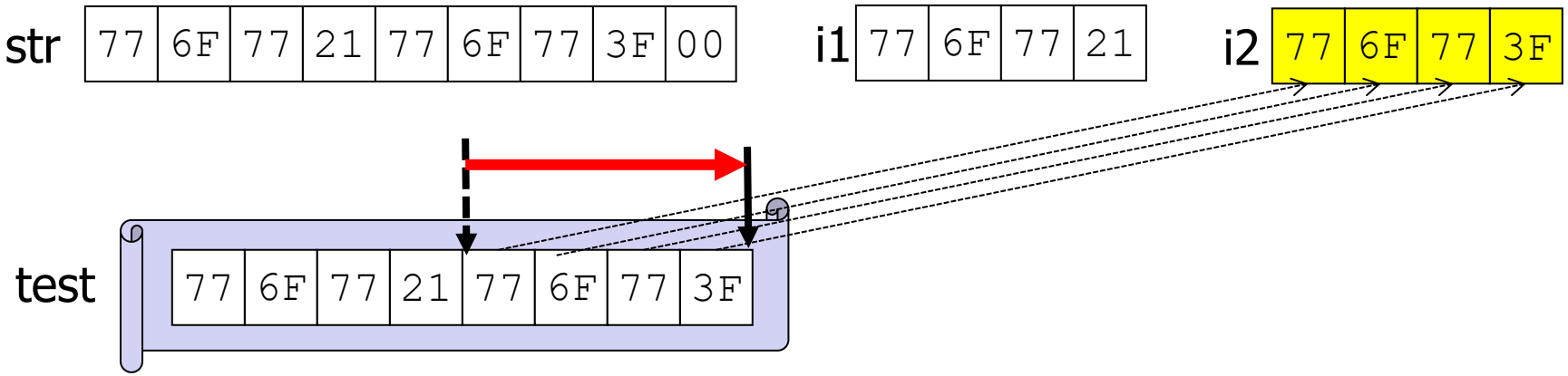
```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```



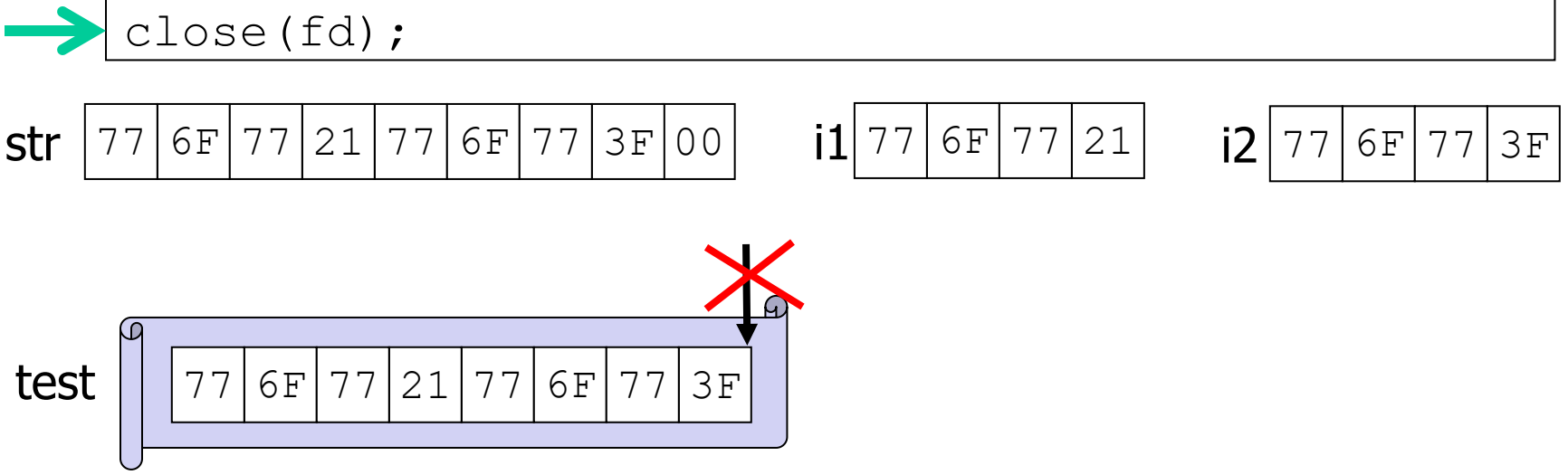
```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```



```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```



```
int fd;
int i1, i2;
char str[]="wow!wow?";

fd = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
write(fd,str,8);
lseek(fd,0,SEEK_SET);
read(fd,&i1,sizeof(int));
read(fd,&i2,sizeof(int));
close(fd);
```

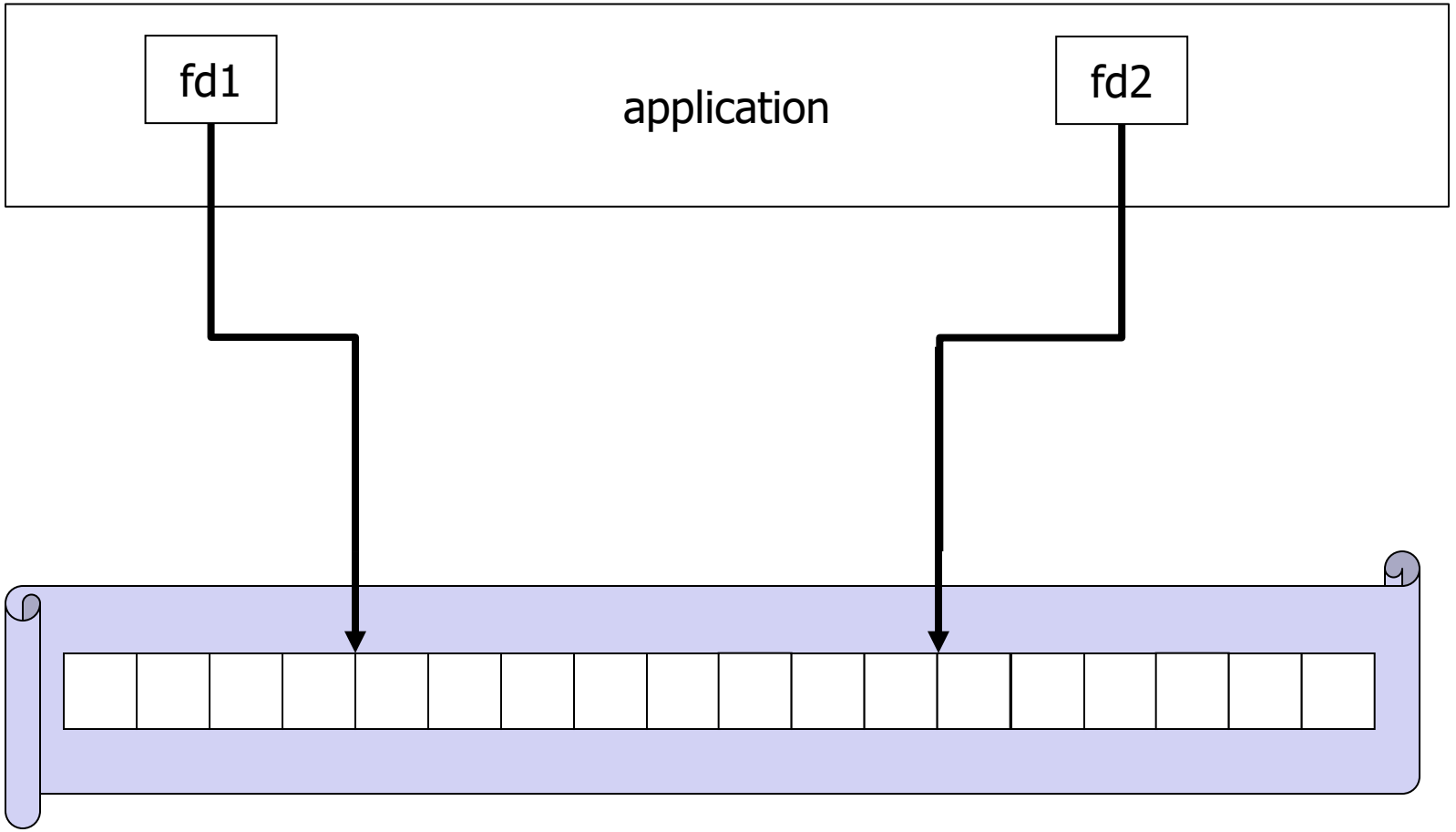


Ξεχωριστοί περιγραφείς στο ίδιο αρχείο

- Ένα πρόγραμμα μπορεί να δημιουργεί/διατηρεί **ξεχωριστούς** περιγραφείς για το **ίδιο** αρχείο

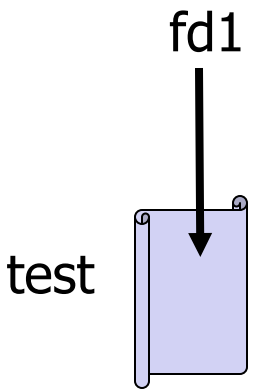
- **Κάθε** περιγραφέας έχει **ξεχωριστή θέση** ανάγνωσης/εγγραφής πάνω στο αρχείο
  - λειτουργίες που εκτελούνται μέσω ενός περιγραφέα **δεν επηρεάζουν** την θέση ανάγνωση/εγγραφής των άλλων περιγραφέων

- Αυτό δίνει μεγαλύτερη ευελιξία
  - π.χ. αν επιθυμούμε να πραγματοποιήσουμε ενέργειες σε διαφορετικά σημεία του αρχείου – αντί να αλλάζουμε συνεχώς την θέση ανάγνωσης/εγγραφής ενός μοναδικού περιγραφέα

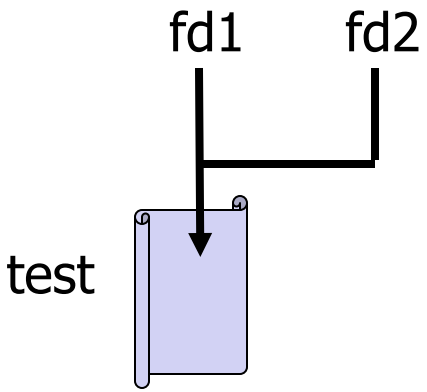

fd1
application
fd2

```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```

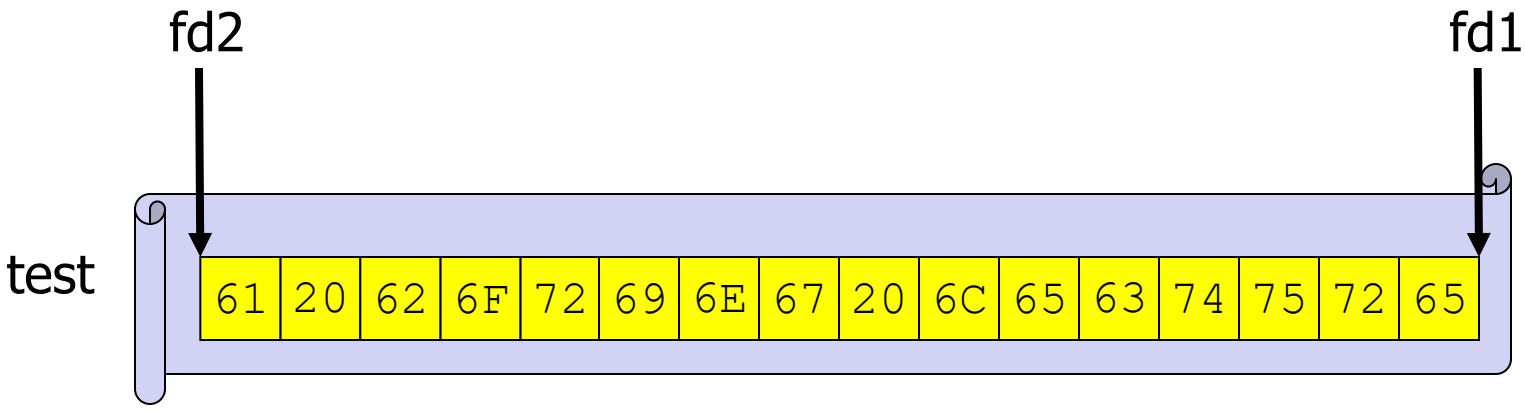
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



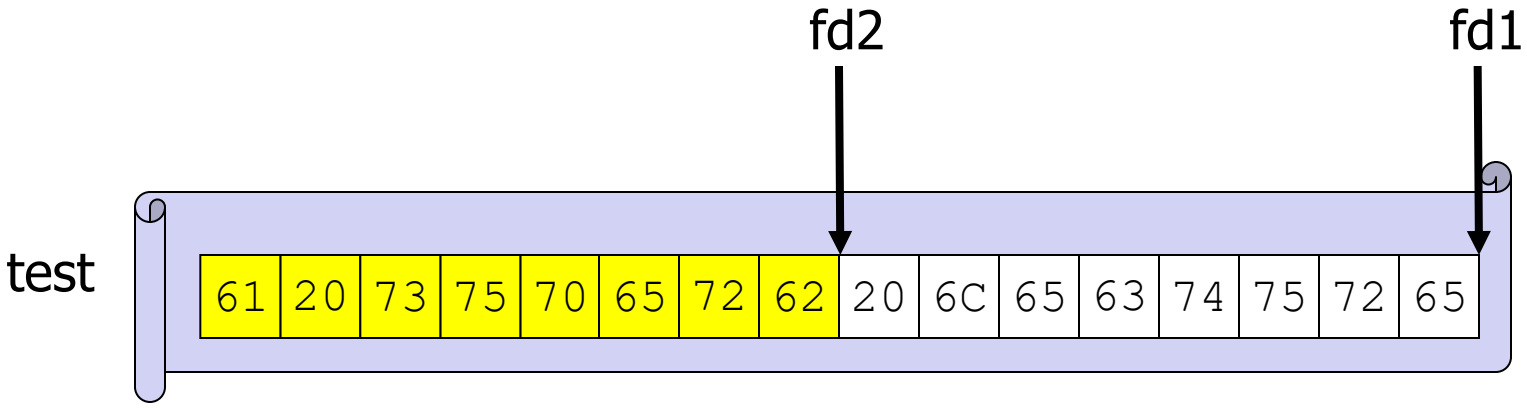
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



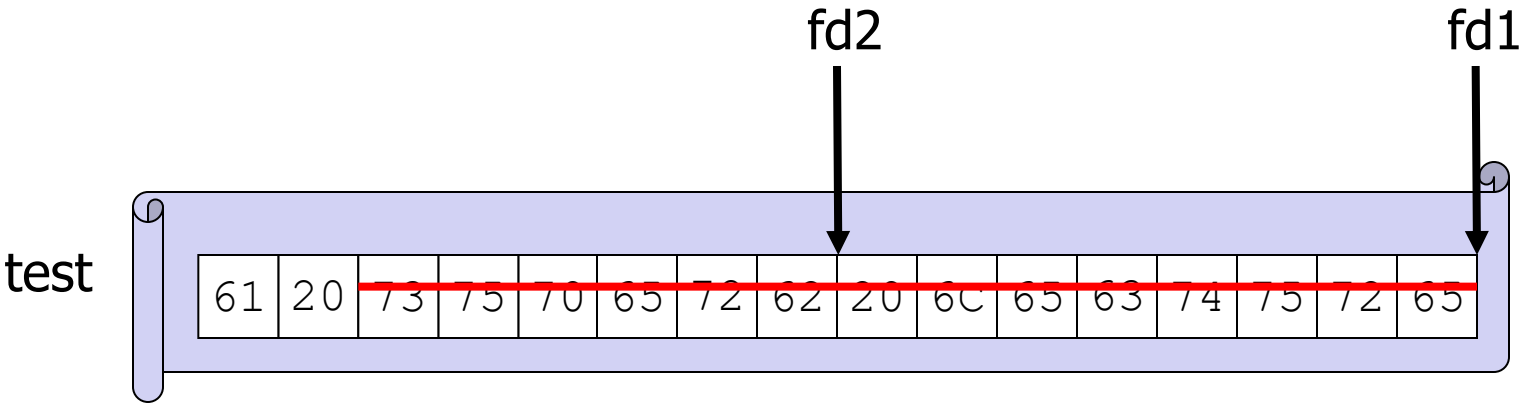
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



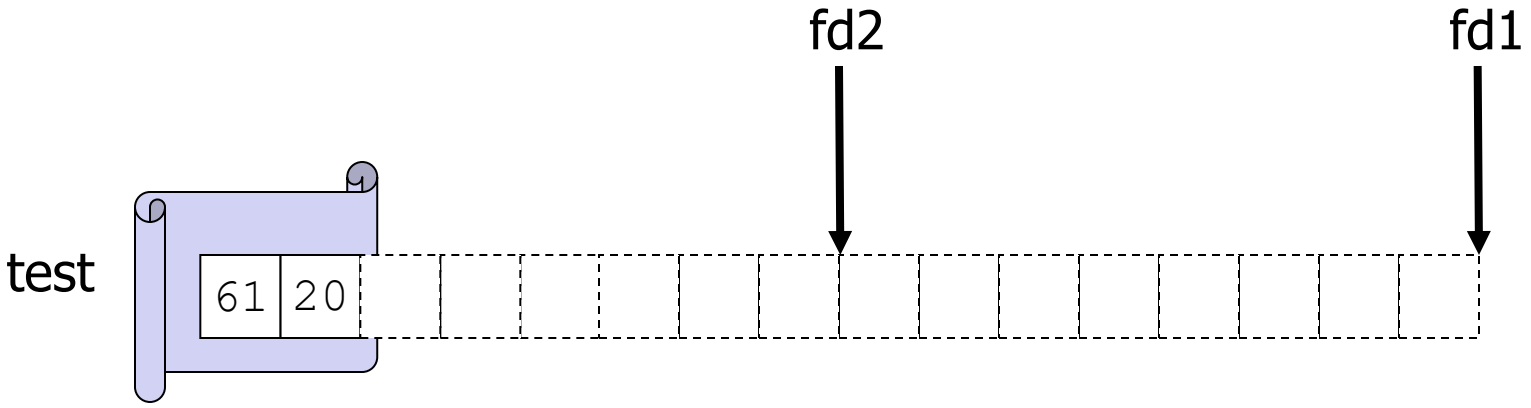
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



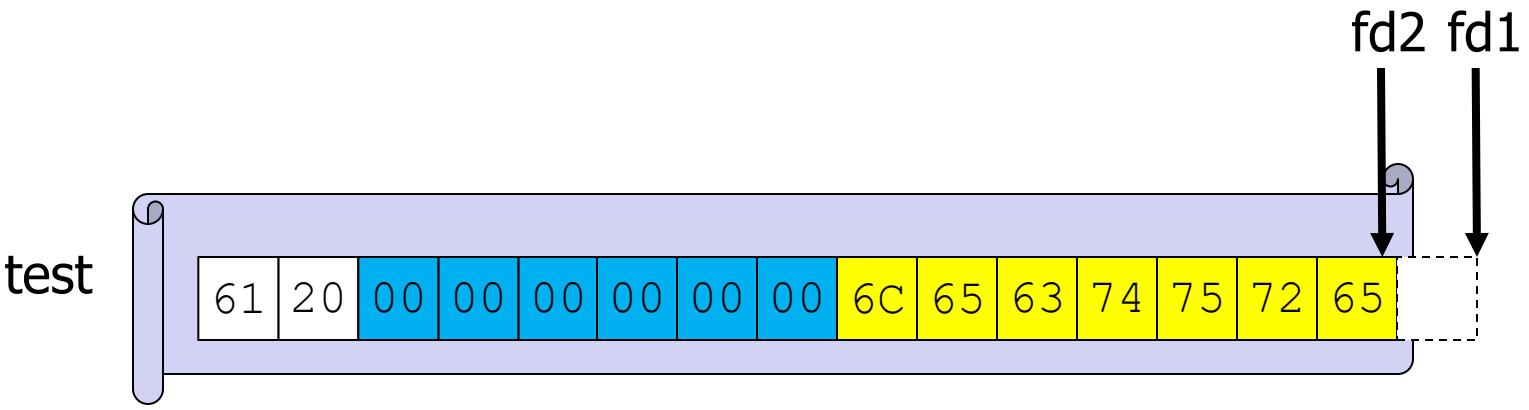
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



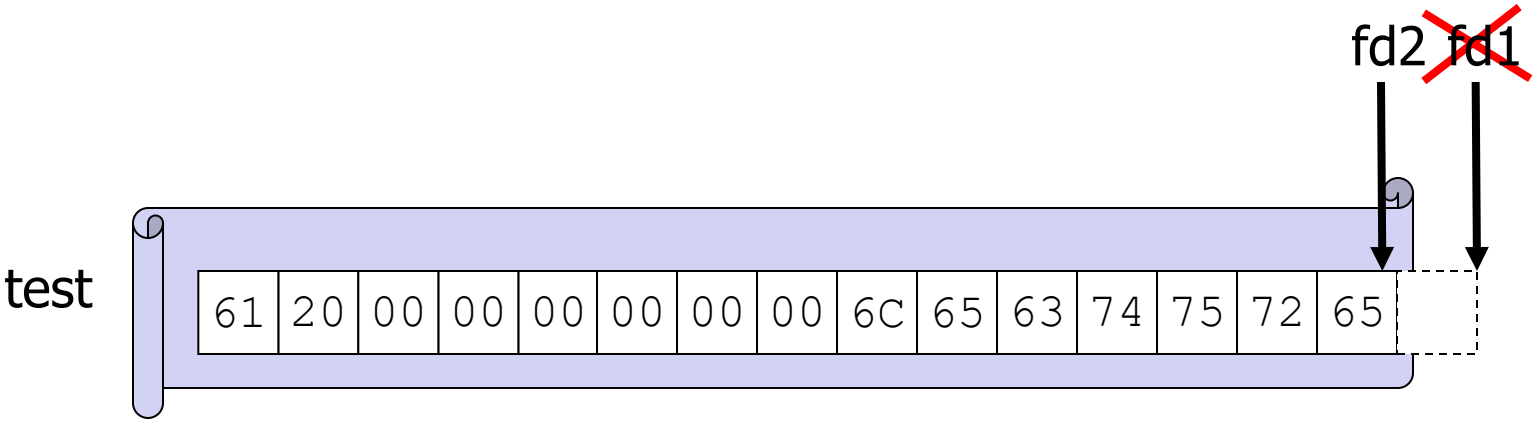
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



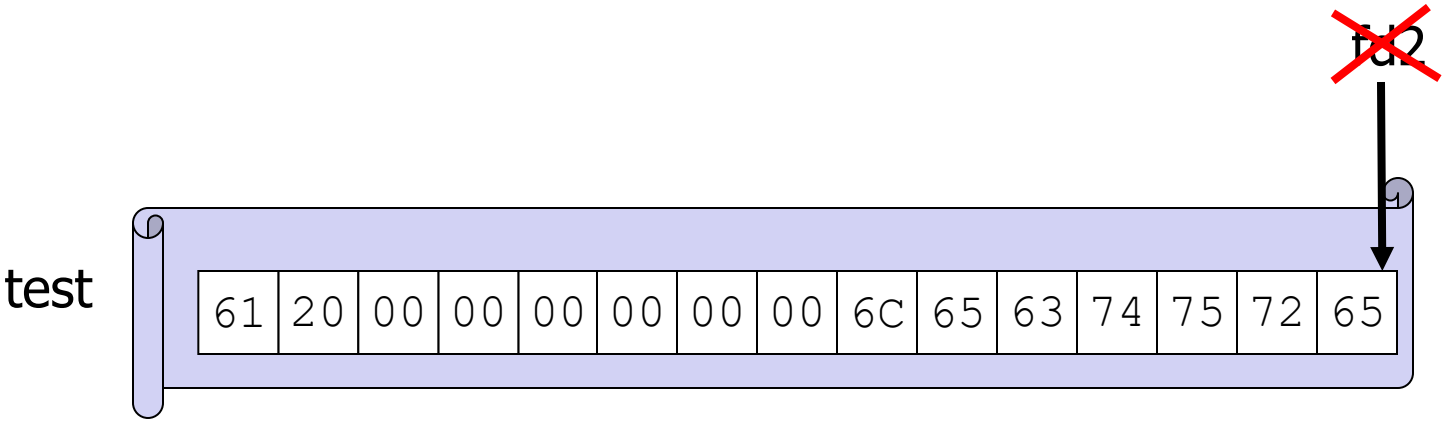
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



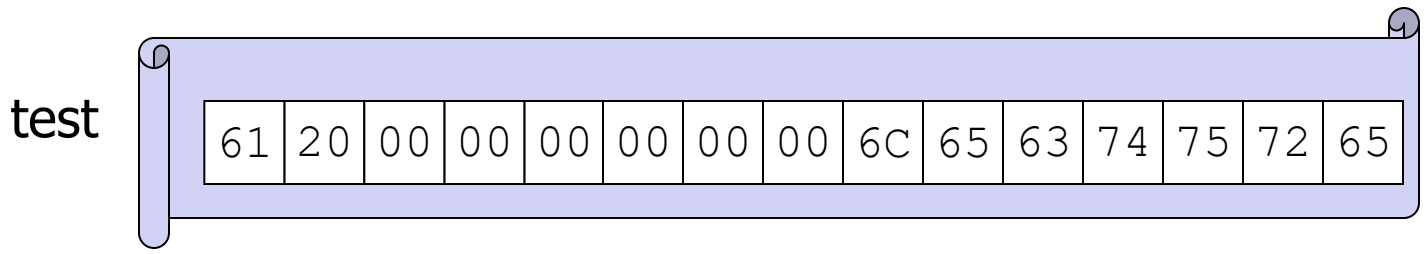
```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



```
int fd1,fd2;

fd1 = open("test",O_RDWR|O_CREAT|O_TRUNC,S_IRWXU);
fd2 = open("test",O_RDWR,0);
write(fd1,"a boring lecture",16);
write(fd2,"a superb",8);
ftruncate(fd1,2);
write(fd2,"lecture",7);
close(fd1);
close(fd2);
```



Λήψη πληροφορίας αρχείου

- Κάθε αρχείο είναι συσχετισμένο με ένα inode που περιέχει διάφορες πληροφορίες
  - τύπος, ιδιοκτήτης, μέγεθος, ώρα πιο πρόσφατης προσπέλασης
- Υπάρχουν ειδικές λειτουργίες για την λήψη αυτής της πληροφορίας

- `int stat(const char *path, struct stat *buf)`
  - λήψη (μετα)πληροφορίας αρχείου με βάση το όνομα/μονοπάτι
- `int fstat(int fd, stuct stat *buf)`
  - λήψη (μετα)πληροφορίας αρχείου με βάση έναν περιγραφέα αρχείου
- Η δομή `struct stat` περιέχει διάφορα πεδία μέσω των οποίων επιστρέφεται η ζητούμενη πληροφορία

```
#include <stdio.h>
#include <sys/stat.h>
#include <time.h>

int main(int argc c, char *argv[]) {
  struct stat sbuf;

  if (stat(argv[1], &sbuf) == -1) {
    perror("stat");
    return(1);
  }

  printf("inode: %ld\n", sbuf.st_ino);
  printf("owner: %d\n", sbuf.st_uid);
  printf("permissions: %o\n", sbuf.st_mode &
                              (S_IRWXU|S_IRWXG|S_IRWXO));
  printf("size: %ld\n", sbuf.st_size);
  printf("last accessed: %s", ctime(&sbuf.st_atime));

  return(0);
}
```

Αποσύνδεση (ονόματος) αρχείου

- `int unlink(const char *path)`

- Καταργεί τον **σύνδεσμο (όνομα)** στο αρχείο
  - μειώνεται ο αριθμός συνδέσμων που δείχνουν στο inode

- Αν δεν υπάρχει άλλος σύνδεσμος σε αυτό το αρχείο, το αρχείο καθίσταται **απροσπέλαστο**
  - οπότε **διαγράφεται** όταν κλείσει ο τελευταίος περιγραφέας σε αυτό