

Per entendre bé la posició on som i la seva relació amb `SEEK_CUR`



- Recordeu que vam parlar de què conté l'estructura `FILE` associada a un arxiu?



Estructura de `FILE`

- **Un buffer de lectura/escriptura:**
- **Un punter a la posició actual dins del fitxer**, que indica on estem llegint o escrivint dins del fitxer.
- **Un descriptor de fitxer:** És un número enter assignat pel sistema operatiu quan s'obre un fitxer.
- **Flags d'estat intern:** Són indicadors que el sistema utilitza per saber si hi ha hagut algun problema amb el fitxer.
 - **Flag EOF (End-Of-File)**
 - **Flag d'error**
- El **mode d'obertura**, que indica com es va obrir el fitxer (lectura, escriptura, mode binari, etc.)



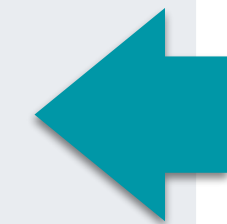
Per entendre bé la posició on som i la seva relació amb `SEEK_CUR`

- Recordeu que vam parlar de què conté l'estructura `FILE` associada a un arxiu?



Estructura de `FILE`

- **Un buffer de lectura/escriptura:**
- **Un punter a la posició actual dins del fitxer**, que indica on estem llegint o escrivint dins del fitxer.
- **Un descriptor de fitxer:** És un número enter assignat pel sistema operatiu quan s'obre un fitxer.
- **Flags d'estat intern:** Són indicadors que el sistema utilitza per saber si hi ha hagut algun problema amb el fitxer.
 - **Flag EOF (End-Of-File)**
 - **Flag d'error**
- El **mode d'obertura**, que indica com es va obrir el fitxer (lectura, escriptura, mode binari, etc.)



Dins de cada estructura `FILE` hi ha un punter a la posició actual on estem llegint (això també passa amb els arxius de text pla).

Per entendre bé la posició on som i la seva relació amb `SEEK_CUR`

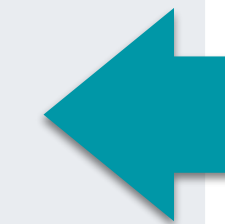


- Recordeu que vam parlar de què conté l'estructura `FILE` associada a un arxiu?



Estructura de `FILE`

- **Un buffer de lectura/escriptura:**
- **Un punter a la posició actual dins del fitxer**, que indica on estem llegint o escrivint dins del fitxer.
- **Un descriptor de fitxer:** És un número enter assignat pel sistema operatiu quan s'obre un fitxer.
- **Flags d'estat intern:** Són indicadors que el sistema utilitza per saber si hi ha hagut algun problema amb el fitxer.
 - **Flag EOF (End-Of-File)**
 - **Flag d'error**
- El **mode d'obertura**, que indica com es va obrir el fitxer (lectura, escriptura, mode binari, etc.)



Dins de cada estructura `FILE` hi ha un punter a la posició actual on estem llegint (això també passa amb els arxius de text pla).

- Aquest punter:
 - Podem modificar-lo amb la funció `fseek()` (com acabem de veure)
 - Podem consultar el seu valor amb la funció `ftell()`

La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int id;
    float valor;
} Parella;
```

```
int main() {
    FILE *f_in = fopen("registres.bin", "rb");
    if (f_in == NULL) {
        fprintf(stderr, "Error en obrir el fitxer");
        exit(EXIT_FAILURE);
    }
    Parella r;
    printf("Un registre Parella ocupa: %d bytes\n", sizeof(Parella));
    // Imprimeixo posicio ara que acabo d'obrir
    long posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);

    fseek(f_in, sizeof(Parella) * 2, SEEK_SET);
    // Imprimeixo posicio ara que m'he mogut
    posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);
    fread(&r, sizeof(Parella), 1, f_in);
    printf("Registre llegit: ID=%d, Valor=%.2f\n", r.id, r.valor);

    fclose(f_in);
    return 0;
}
```


La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int id;
    float valor;
} Parella;
```

```
int main() {
    FILE *f_in = fopen("registres.bin", "rb");
    if (f_in == NULL) {
        fprintf(stderr, "Error en obrir el fitxer");
        exit(EXIT_FAILURE);
    }
    Parella r;
    printf("Un registre Parella ocupa: %d bytes\n", sizeof(Parella));
    // Imprimeixo posicio ara que acabo d'obrir
    long posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);

    fseek(f_in, sizeof(Parella) * 2, SEEK_SET);
    // Imprimeixo posicio ara que m'he mogut
    posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);
    fread(&r, sizeof(Parella), 1, f_in);
    printf("Registre llegit: ID=%d, Valor=%.2f\n", r.id, r.valor);

    fclose(f_in);
    return 0;
}
```

Imprimeixo la mida de Parella per poder entendre després quant valen els desplaçaments

Un registre Parella ocupa: 8 bytes

La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int id;
    float valor;
} Parella;
```

```
int main() {
    FILE *f_in = fopen("registres.bin", "rb");
    if (f_in == NULL) {
        fprintf(stderr, "Error en obrir el fitxer");
        exit(EXIT_FAILURE);
    }
    Parella r;
    printf("Un registre Parella ocupa: %d bytes\n", sizeof(Parella));
    // Imprimeixo posicio ara que acabo d'obrir
    long posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);

    fseek(f_in, sizeof(Parella) * 2, SEEK_SET);
    // Imprimeixo posicio ara que m'he mogut
    posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);
    fread(&r, sizeof(Parella), 1, f_in);
    printf("Registre llegit: ID=%d, Valor=%.2f\n", r.id, r.valor);

    fclose(f_in);
    return 0;
}
```

Calculo la posició amb `ftell()` ara que acabo d'obrir. Hauria de ser 0 bytes perquè estic a principi de fitxer.

Un registre Parella ocupa: 8 bytes

La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int id;
    float valor;
} Parella;
```

```
int main() {
    FILE *f_in = fopen("registres.bin", "rb");
    if (f_in == NULL) {
        fprintf(stderr, "Error en obrir el fitxer");
        exit(EXIT_FAILURE);
    }
    Parella r;
    printf("Un registre Parella ocupa: %d bytes\n", sizeof(Parella));
    // Imprimeixo posicio ara que acabo d'obrir
    long posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);

    fseek(f_in, sizeof(Parella) * 2, SEEK_SET);
    // Imprimeixo posicio ara que m'he mogut
    posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);
    fread(&r, sizeof(Parella), 1, f_in);
    printf("Registre llegit: ID=%d, Valor=%.2f\n", r.id, r.valor);

    fclose(f_in);
    return 0;
}
```

Calculo la posició amb `ftell()` ara que acabo d'obrir. Hauria de ser 0 bytes perquè estic a principi de fitxer.

Un registre Parella ocupa: 8 bytes
Posicio actual: 0 ← OK

La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int id;
    float valor;
} Parella;
```

```
int main() {
    FILE *f_in = fopen("registres.bin", "rb");
    if (f_in == NULL) {
        fprintf(stderr, "Error en obrir el fitxer");
        exit(EXIT_FAILURE);
    }
    Parella r;
    printf("Un registre Parella ocupa: %d bytes\n", sizeof(Parella));
    // Imprimeixo posicio ara que acabo d'obrir
    long posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);

    fseek(f_in, sizeof(Parella) * 2, SEEK_SET);
    // Imprimeixo posicio ara que m'he mogut
    posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);
    fread(&r, sizeof(Parella), 1, f_in);
    printf("Registre llegit: ID=%d, Valor=%.2f\n", r.id, r.valor);

    fclose(f_in);
    return 0;
}
```

Després de moure'm 2 registres torno a imprimir la posicio actual. Si una parella ocupava 8 B, hauria d'estar a la posició 16.

Un registre Parella ocupa: 8 bytes
Posicio actual: 0

La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int id;
    float valor;
} Parella;
```

```
int main() {
    FILE *f_in = fopen("registres.bin", "rb");
    if (f_in == NULL) {
        fprintf(stderr, "Error en obrir el fitxer");
        exit(EXIT_FAILURE);
    }
    Parella r;
    printf("Un registre Parella ocupa: %d bytes\n", sizeof(Parella));
    // Imprimeixo posicio ara que acabo d'obrir
    long posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);

    fseek(f_in, sizeof(Parella) * 2, SEEK_SET);
    // Imprimeixo posicio ara que m'he mogut
    posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);
    fread(&r, sizeof(Parella), 1, f_in);
    printf("Registre llegit: ID=%d, Valor=%.2f\n", r.id, r.valor);

    fclose(f_in);
    return 0;
}
```

Després de moure'm 2 registres torno a imprimir la posicio actual. Si una parella ocupava 8 B, hauria d'estar a la posició 16.

Un registre Parella ocupa: 8 bytes
Posicio actual: 0
Posicio actual: 16 ← OK

La funció `ftell()`

- La funció `ftell()` rep per paràmetre una estructura `file` i ens retorna la posició actual, dins d'aquell fitxer.

```
long ftell(FILE *fitxer);
```

- "La posició actual" és la distància en bytes des del començament del fitxer fins a on es troba actualment el punter de lectura/escriptura.
- Fem servir l'exemple anterior per veure com funciona:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int id;
    float valor;
} Parella;
```

```
int main() {
    FILE *f_in = fopen("registres.bin", "rb");
    if (f_in == NULL) {
        fprintf(stderr, "Error en obrir el fitxer");
        exit(EXIT_FAILURE);
    }
    Parella r;
    printf("Un registre Parella ocupa: %d bytes\n", sizeof(Parella));
    // Imprimeixo posicio ara que acabo d'obrir
    long posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);

    fseek(f_in, sizeof(Parella) * 2, SEEK_SET);
    // Imprimeixo posicio ara que m'he mogut
    posicio = ftell(f_in);
    printf("Posicio actual: %ld\n", posicio);
    fread(&r, sizeof(Parella), 1, f_in);
    printf("Registre llegit: ID=%d, Valor=%.2f\n", r.id, r.valor);

    fclose(f_in);
    return 0;
}
```

Després de moure'm 2 registres torno a imprimir la posicio actual. Si una parella ocupava 8 B, hauria d'estar a la posició 16.

```
Un registre Parella ocupa: 8 bytes
Posicio actual: 0
Posicio actual: 16 ← OK
Registre llegit: ID=3, Valor=1.61
```

SEEK_SET, SEEK_CUR i SEEK_END

Fins ara hem entès que:

- La posició actual de "on som al fitxer" és una dada que es mesura en bytes des de l'inici i que es guarda dins l'estructura `FILE`
- Que podem consultar-la amb la funció `ftell()`

La pregunta ara és com ho fa `fseek()` per moure's fent servir `SEEK_SET`, `SEEK_CUR` i `SEEK_END`

SEEK_SET, SEEK_CUR i SEEK_END

Fins ara hem entès que:

- La posició actual de "on som al fitxer" és una dada que es mesura en bytes des de l'inici i que es guarda dins l'estructura `FILE`
- Que podem consultar-la amb la funció `ftell()`

La pregunta ara és com ho fa `fseek()` per moure's fent servir `SEEK_SET`, `SEEK_CUR` i `SEEK_END`



Què són `SEEK_SET`, `SEEK_CUR` i `SEEK_END`?

- No són res més que macros definides dins de `stdlib.h`, i valen:

```
#define SEEK_SET    0    /* set file offset to offset */
#define SEEK_CUR    1    /* set file offset to current plus offset */
#define SEEK_END    2    /* set file offset to EOF plus offset */
```

- Per tant, només són una manera curta de dir-li a `fseek` des d'on vols que calculi el desplaçament. Recordar què volen dir 0, 1 o 2 seria un malson.

SEEK_SET, SEEK_CUR i SEEK_END

?

Què fa internament `fseek()`?

Recorda la capçalera: `int fseek(FILE *fitxer, long desplaçament, int origen);`

SEEK_SET, SEEK_CUR i SEEK_END

?

Què fa internament `fseek()`?

Recorda la capçalera: `int fseek(FILE *fitxer, long desplaçament, int origen);`

1

Llegeix la posició actual del punter (`posicio_actual`) (només en el cas de `SEEK_CUR`, per la resta, no cal)

SEEK_SET, SEEK_CUR i SEEK_END

?

Què fa internament `fseek()`?

Recorda la capçalera: `int fseek(FILE *fitxer, long desplaçament, int origen);`

1 Llegeix la posició actual del punter (`posicio_actual`) (només en el cas de `SEEK_CUR`, per la resta, no cal)

2 Calcula la nova posició absoluta (`nova_posicio`) segons el valor d'origen:

Si `origen` és `SEEK_SET` (0), `nova_posicio` = `desplaçament`

Si `origen` és `SEEK_CUR` (1), `nova_posicio` = `posicio_actual` + `desplaçament`

Si `origen` és `SEEK_END` (2), `nova_posicio` = `mida_del_fitxer` + `desplaçament`

SEEK_SET, SEEK_CUR i SEEK_END

?

Què fa internament `fseek()`?

Recorda la capçalera: `int fseek(FILE *fitxer, long desplaçament, int origen);`

- 1 Llegeix la posició actual del punter (`posicio_actual`) (només en el cas de `SEEK_CUR`, per la resta, no cal)
- 2 Calcula la nova posició absoluta (`nova_posicio`) segons el valor d'origen:

Si `origen` és `SEEK_SET` (0), `nova_posicio` = `desplaçament`

Si `origen` és `SEEK_CUR` (1), `nova_posicio` = `posicio_actual` + `desplaçament` *Ha de ser negatiu!*

Si `origen` és `SEEK_END` (2), `nova_posicio` = `mida_del_fitxer` + `desplaçament`

SEEK_SET, SEEK_CUR i SEEK_END

?

Què fa internament `fseek()`?

Recorda la capçalera: `int fseek(FILE *fitxer, long desplaçament, int origen);`

1 Llegeix la posició actual del punter (`posicio_actual`) (només en el cas de `SEEK_CUR`, per la resta, no cal)

2 Calcula la nova posició absoluta (`nova_posicio`) segons el valor d'origen:

Si `origen` és `SEEK_SET` (0), `nova_posicio` = `desplaçament`

Si `origen` és `SEEK_CUR` (1), `nova_posicio` = `posicio_actual` + `desplaçament` *Ha de ser negatiu!*

Si `origen` és `SEEK_END` (2), `nova_posicio` = `mida_del_fitxer` + `desplaçament`

3 Comprova que la nova posició sigui vàlida (no negativa). Si és negativa retorna un codi d'error (-1) i surt.


SEEK_SET, SEEK_CUR i SEEK_END

?

Què fa internament `fseek()`?

Recorda la capçalera: `int fseek(FILE *fitxer, long desplaçament, int origen);`

- 1 Llegeix la posició actual del punter (`posicio_actual`) (només en el cas de `SEEK_CUR`, per la resta, no cal)
- 2 Calcula la nova posició absoluta (`nova_posicio`) segons el valor d'origen:

Si `origen` és `SEEK_SET` (0), `nova_posicio` = `desplaçament`
Si `origen` és `SEEK_CUR` (1), `nova_posicio` = `posicio_actual` + `desplaçament`
Si `origen` és `SEEK_END` (2), `nova_posicio` = `mida_del_fitxer` + `desplaçament`  *Ha de ser negatiu!*
- 3 Comprova que la nova posició sigui vàlida (no negativa). Si és negativa retorna un codi d'error (-1) i surt.
- 4 Fa una crida al sistema operatiu per posicionar físicament el punter del fitxer.

SEEK_SET, SEEK_CUR i SEEK_END

?


Què fa internament `fseek()`?

Recorda la capçalera: `int fseek(FILE *fitxer, long desplaçament, int origen);`

- 1 Llegeix la posició actual del punter (`posicio_actual`) (només en el cas de `SEEK_CUR`, per la resta, no cal)
- 2 Calcula la nova posició absoluta (`nova_posicio`) segons el valor d'origen:

Si `origen` és `SEEK_SET` (0), `nova_posicio` = `desplaçament`

Si `origen` és `SEEK_CUR` (1), `nova_posicio` = `posicio_actual` + `desplaçament` *Ha de ser negatiu!*

Si `origen` és `SEEK_END` (2), `nova_posicio` = `mida_del_fitxer` + `desplaçament` 
- 3 Comprova que la nova posició sigui vàlida (no negativa). Si és negativa retorna un codi d'error (-1) i surt.
- 4 Fa una crida al sistema operatiu per posicionar físicament el punter del fitxer.
- 5 Actualitza el punter intern de l'estructura `FILE` amb el nou valor.

També pots modificar-lo!



- I si vull canviar el valor d'aquest tercer registre? Vull que el camp `valor` valgui 666 en comptes de 1.61

Instruccions:

1. Obrir en mode lectura i escriptura

Com que hem de llegir-lo i modificar-lo al mateix fitxer, necessitem obrir en mode de lectura i escriptura "rb"

```
FILE *fit = fopen("registres.bin", "rb+");
```

2. Busquem el tercer registre i el llegim

Per modificar-lo, primer l'hem de llegir. En aquest cas el guardem en una variable anomenada `r`.

```
fseek(fit, sizeof(Parella) * 2, SEEK_SET);  
fread(&r, sizeof(Parella), 1, fit);
```

3. En modifiquem el valor

```
r.valor = 666;
```

4. Tornar a buscar la posició i escriure'l

Ara ja el podem escriure, però com que abans hem fet un `fread()`, no estem a la posició correcta. L'hem de tornar a buscar.

```
fseek(fit, sizeof(Parella) * 2, SEEK_SET);  
fwrite(&r, sizeof(Parella), 1, fit);
```

5. Flush o close

Si ja haguéssim acabat, tancaríem el fitxer. Però com que volem comprovar que s'hagi guardat bé des d'aquest mateix programa, forcem que s'escriguin els continguts al fitxer.

```
fflush(fit);
```

6. Tornar a llegir per comprovar que s'ha modificat

Altre cop, necessitem buscar la posició des de l'inici. Ara el guardo en una altra variable de tipus `Parella`, anomenada `p`.

```
fseek(f_in, sizeof(Parella) * 2, SEEK_SET);  
fread(&p, sizeof(Parella), 1, f_in);
```

➡ Tens l'exemple complet al moodle,
anomenat `modificar_registre_fseek.c`

Una altra manera de trobar la posició: tirant enrere



- En comptes de tornar al principi, podem quedar-nos on érem i tirar enrere, des de `SEEK_CUR`? Efectivament!



Una altra manera de trobar la posició: tirant enrere

- En comptes de tornar al principi, podem quedar-nos on érem i tirar enrere, des de `SEEK_CUR`? Efectivament!

L'únic que hem de fer diferent és aquest pas



Instruccions:

1. Obrir en mode lectura i escriptura

Com que hem de llegir-lo i modificar-lo al mateix fitxer, necessitem obrir en mode de lectura i escriptura "rb"

```
FILE *fit = fopen("registres.bin", "rb+");
```

2. Busquem el tercer registre i el llegim

Per modificar-lo, primer l'hem de llegir. En aquest cas el guardem en una variable anomenada `r`.

```
fseek(fit, sizeof(Parella) * 2, SEEK_SET);  
fread(&r, sizeof(Parella), 1, fit);
```

3. En modifiquem el valor

```
r.valor = 666;
```

4. Tornar a la posició que toca (tirar enrere)

Per tirar enrere ens hem de desplaçar, respecte la posició actual, el que ocupa un registre. Desplaçament negatiu per tirar enrere.

```
fseek(fit, - sizeof(Parella), SEEK_CUR);  
fwrite(&r, sizeof(Parella), 1, fit);
```

5. Flush o close

Si ja haguéssim acabat, tancaríem el fitxer. Però com que volem comprovar que s'hagi guardat bé des d'aquest mateix programa, forcem que s'escriguin els continguts al fitxer.

```
fflush(fit);
```

6. Tornar a llegir per comprovar que s'ha modificat

Altre cop, necessitem buscar la posició des de l'inici. Ara el guardo en una altra variable de tipus `Parella`, anomenada `p`.

```
fseek(f_in, sizeof(Parella) * 2, SEEK_SET);  
fread(&p, sizeof(Parella), 1, f_in);
```