Ficheiros

sumário

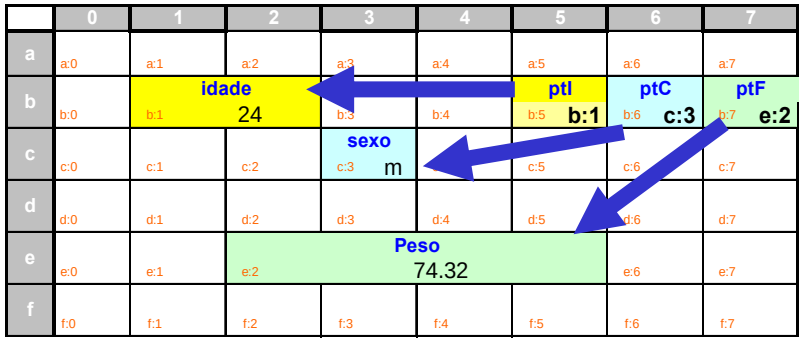
- Revisões sobre ponteiros
- Abrir e Fechar ficheiros
- leitura de caracteres de ficheiros
- Escrita de caracteres em ficheiros
- Leitura e Escrita formatada em ficheiros
- ficheiros definidos em <stdio.h>
- Ficheiros binários
- Resumo

Apontadores em C

revisões

Ponteiros

```
<tipo_variavel> *<nome>;
<tipo_variavel> *<nome> = endereço;
```



Declaração

```
int idade;
char sexo;
float peso;
int *ptI;
char *ptC;
float *ptF;
```

Atribuição

```
idade =24;
sexo = 'm';
peso=74.32;
*ptI = &idade;
*ptC = &sexo;
*ptF = &peso;
```

dados

```
idade = 18;
*pti = 18;
```

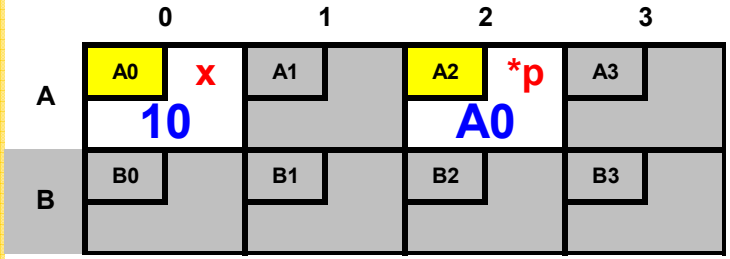
O operador **&** devolve o **endereço** de uma variável
O **nome** de uma variável devolve o seu **conteúdo**

O operador ***** devolve o **conteúdo apontado** por um apontador

Exemplo

Ponteiros e variáveis

```
#include <stdio.h>
int main(int argc , char *argv[])
{
        int x;
        int *p;
       p = &x;
      printf( “introduza x :);
       scanf( "%d", &x );
       printf( “valor introduzido %d\n", *p );
       printf( “posicao de memoria %p ", p );
}
```



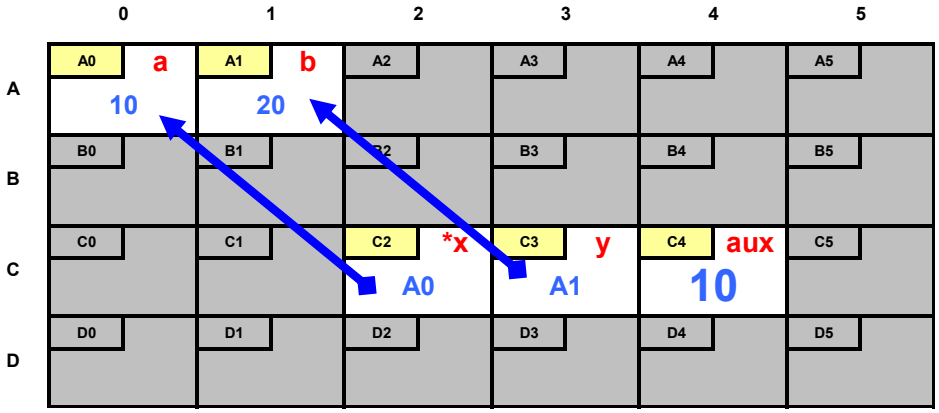

```
introduza x :20
valor introduzido          20
posicao de memoria         0012FF88
```

Passagem de parâmetros por ponteiro

- Permite alterar o conteúdo de um parâmetro
- Evita a cópia dos parâmetros para variáveis locais

Função troca

```
void troca(int *x, int *y)
{
  int aux= *x;
  *x= *y;
  *y=aux;
}
```



Programa

```
. . .
int a= 20;
int b= 10;
troca (&a, &b);
Printf(" a= %d  b=%d ",a, b);
```

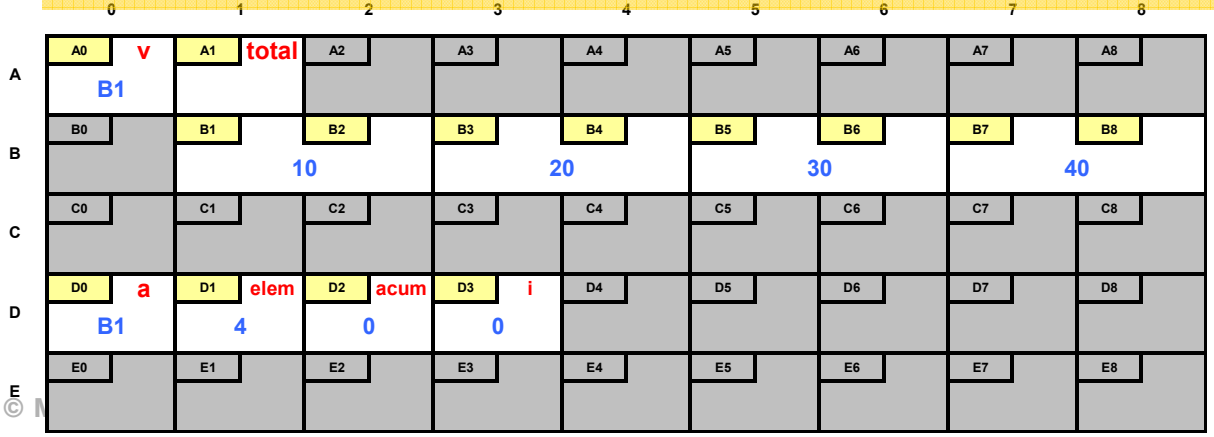
Apontadores e vectores

Programa

```
int v[4] = {10 , 20, 30, 40 };
int total = soma( v , 4 );
```

Funçoes

```
Int soma( int *a , int elem){
  int acum =0, i;
  for( i=0 ; i< elem ; i++)
    acum += a + i;
  return acum;
}
```



NOTA:

O nome de um vector corresponde ao endereço do primeiro dos seus elementos

NOTA:
Nos parâmetros das funções
não é necessário o & porque o vector é um ponteiro

Alocar Memória

void * malloc(numero de bytes)

Devolve um apontador para a memória reservada

int sizeof(tipo)

Calcula o número de bytes que o para metro ocupa

free (apontador)

destruir variáveis

```
int *ptrI= (int *) malloc( sizeof(int) );
char *ptrc = (char *) malloc(sizeof(char) );
double *ptrd = (double *) malloc(sizeof(double) );
. . . .
free(ptri);
free(ptrc);
free(ptrd);
```

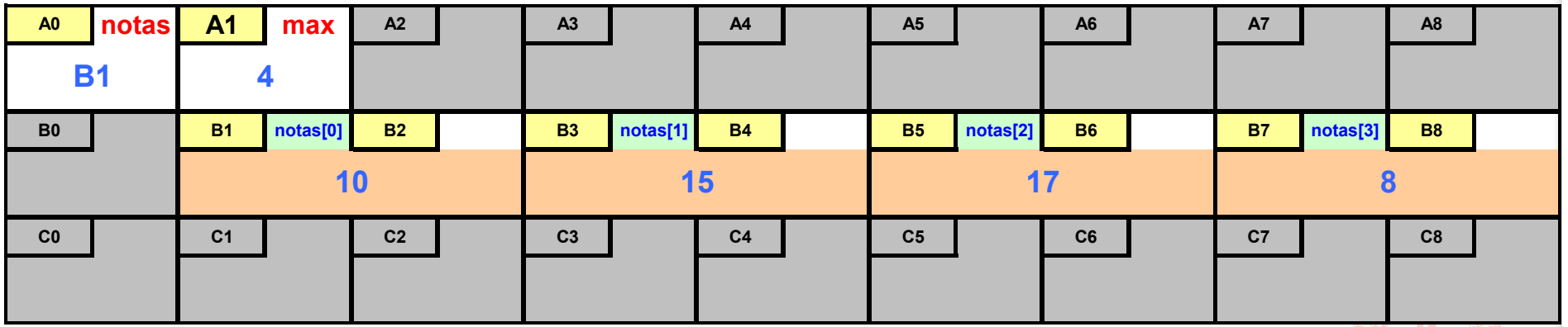
NOTA:

A memória alocada com o malloc deve ser libertada pelo free

Vectores e apontadores

Apontadores com memória própria

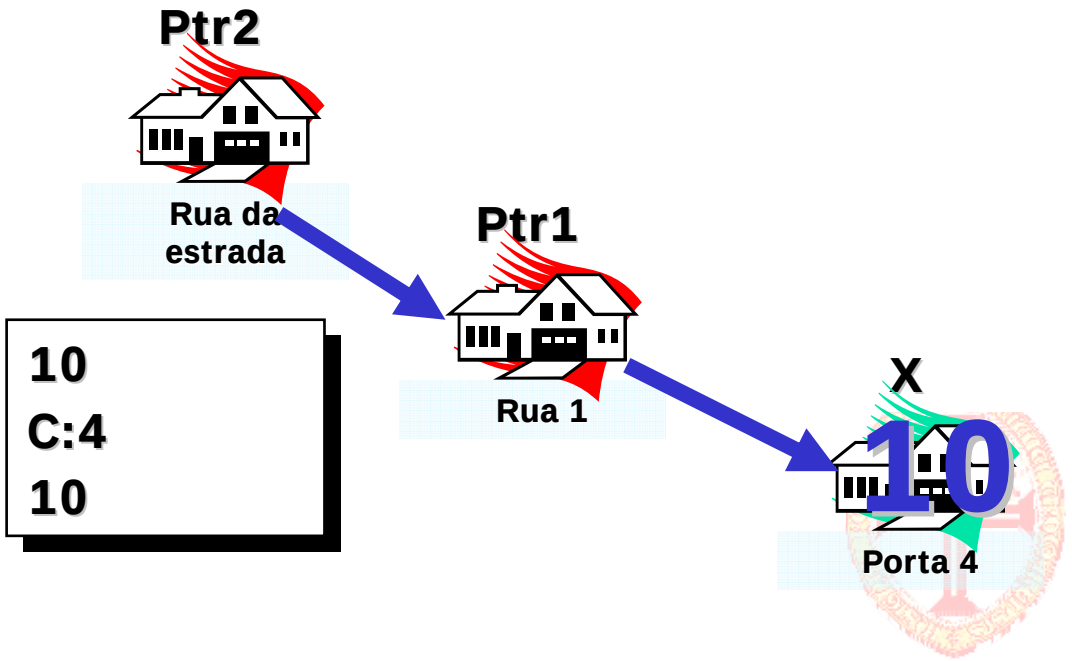
```
int *notas, max;
Scanf("%d",&max);
notas = (int *) malloc( sizeof(int) * max );
. . .
free(notas);
```



Apontadores de Apontadores

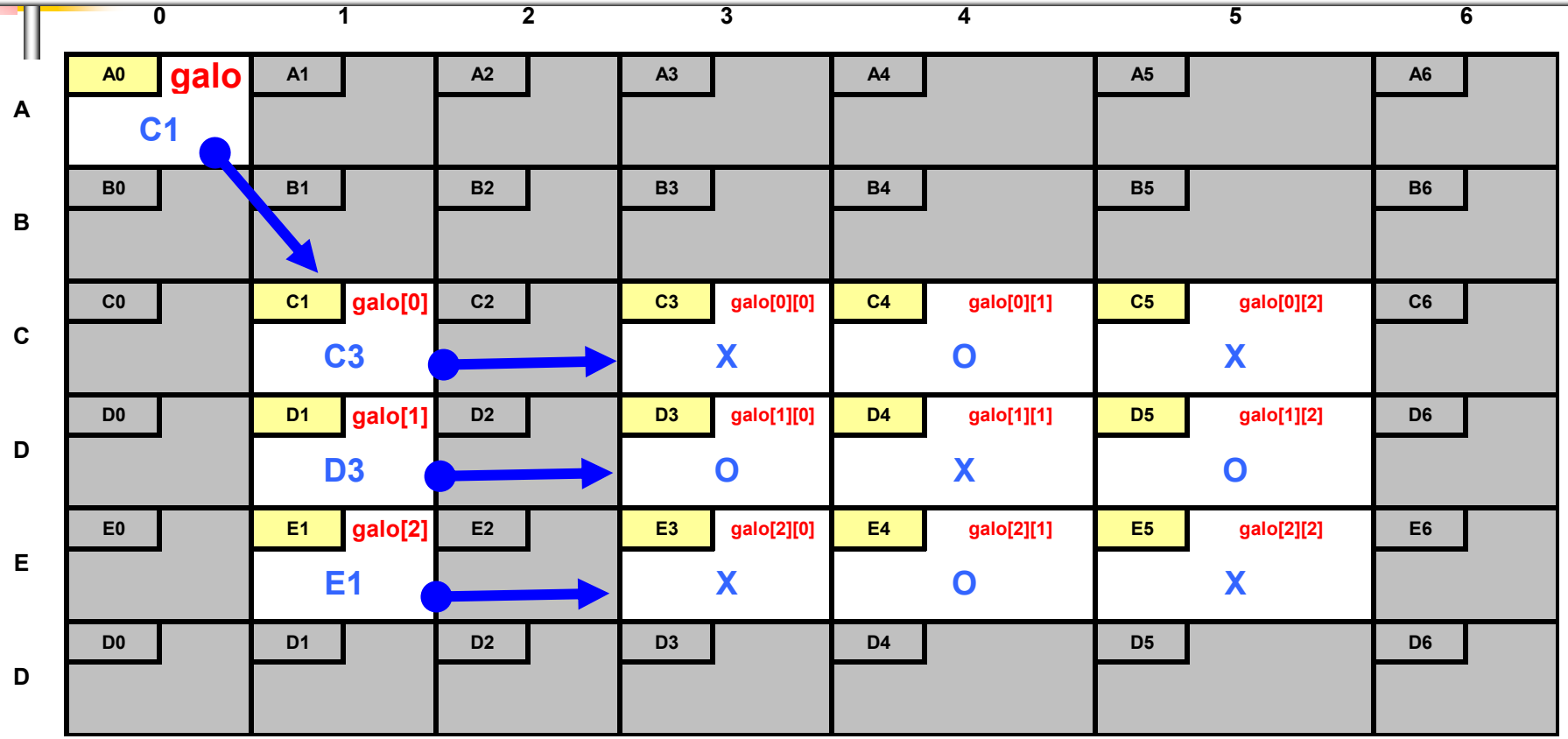
	0	1	2	3	4	5	6	7	8
a	a:0	ptr2 a:1 c:4		a:3	a:4	a:5	a:6	a:7	a:8
b	b:0	b:1	b:2	b:3	b:4	b:5	b:6	b:7	b:8
c	c:0	c:1	c:2	c:3	ptr1 c:4 d:7		c:6	c:7	c:8
d	d:0	d:1	d:2	d:3	d:4	d:5	d:6	x d:7	10

```
Int x=10
Int *ptr1= &x
Int **ptr2 = &ptr1
Printf("%d", *ptr1);
Printf("%p", *ptr2 );
Printf("%d", **ptr2 );
```



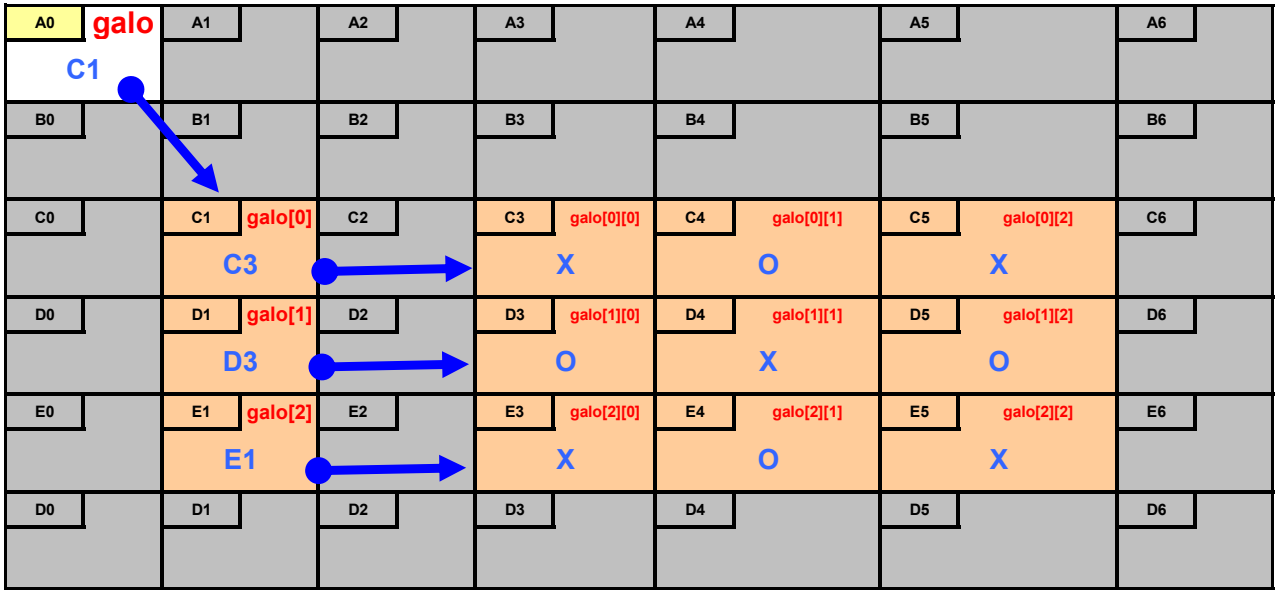

```
10
C:4
10
```

Arrays quadrados




```
Char ** galo;
galo = (char **) malloc( 3 * sizeof(char) );
for( y=0; y < 3;y++)
    galo[y] = (char *) malloc( 3 * sizeof(char) );
```

Arrays quadrados




```
for( y=0; y < dim ; y++)
        free( galo[y] );
free( galo);
```

Exercícios

- Qual a intrução que declara um ponteiro chamado a?
  - A. int a;
  - B. int &a;
  - C. ptr a;
  - D. int *a;
- Qual a instrução que devolve o endereço da variável a ?
  - A. *a;
  - B. a;
  - C. &a;
  - D. address(a);
- Qual a instrução que devolve o endereço da variável que é apontada por a?
  - A. *a;
  - B. a;
  - C. &a;
  - D. address(a);
- Qual a instrução que devolve o valor da variável que é apontada por a?
  - A. a;
  - B. val(a);
  - C. *a;
  - D. &a;
- Quais das seguintes instruções alocam um vector de 10 inteiros
  - A. int a = malloc(10);
  - B. int *a = (int) malloc(40);
  - C. int *a = (int *) malloc(sizeof(int));
  - D. int a = malloc(sizeof(int));
- Quais das seguintes instruções libertam a memória de um vector de 10 inteiros chamada a
  - A. delete a;
  - B. delete (a);
  - C . free (a, 10);
  - D . free(a);

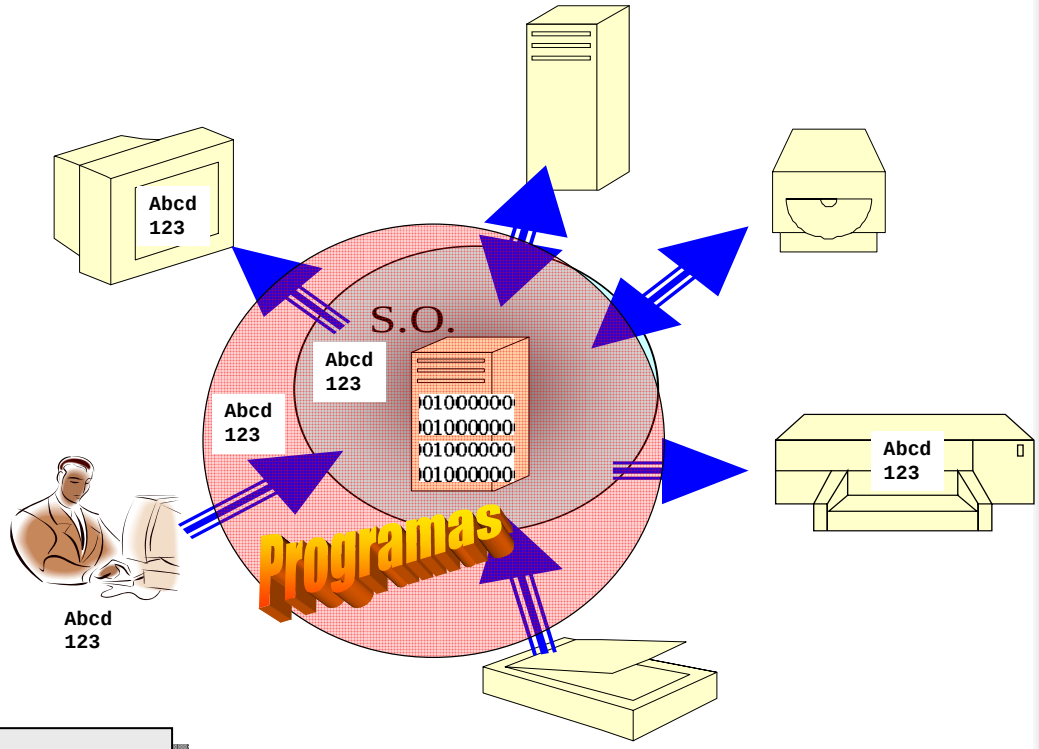
Ficheiros

Memória permanente

Streams (fluxos)

- Streams de entrada
  - Leitura de dados
    - Teclado
    - ficheiros
- Streams de saída
  - Escrita de dados
    - Monitor
    - Impressora
    - ficheiros
- Streams Entrada/Saída
  - Leitura e escrita de dados
    - ficheiros

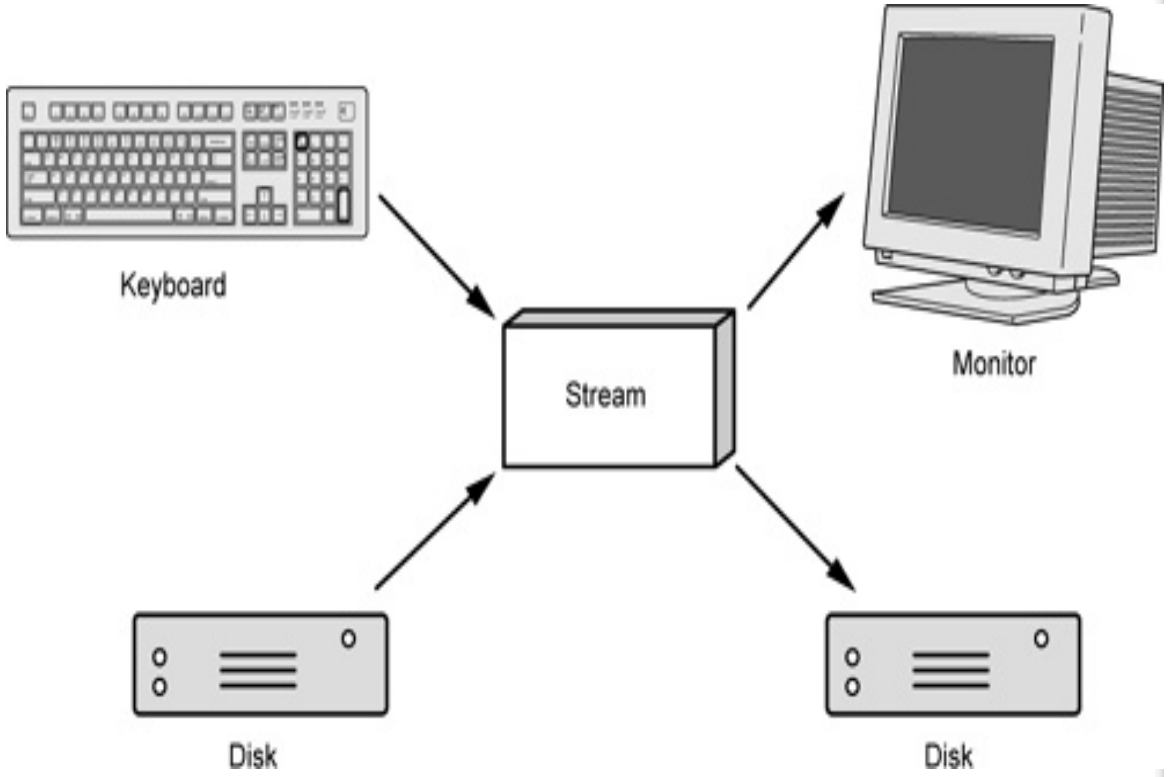
As streams são "Device Independent"

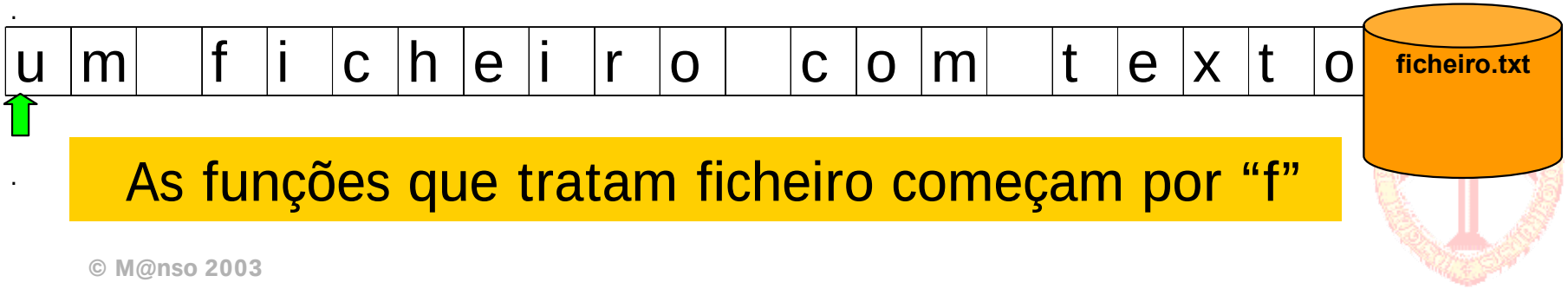


As streams não representam somente ficheiros mas dispositivos I/O

Operação básicas sobre ficheiros

- Abrir a stream
  - fopen
- Ler dados
  - fscanf
  - fgetc
- Escrever dados
  - fprintf
  - fputc
- Fechar a stream
  - fclose

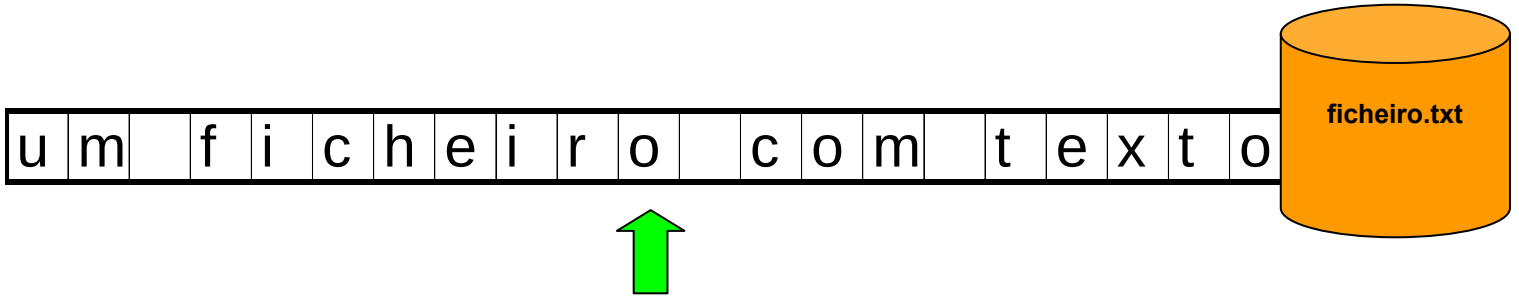




As funções que tratam ficheiro começam por “f”

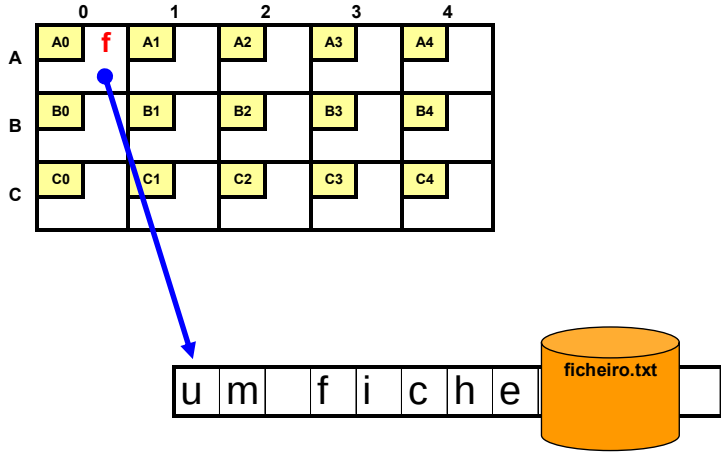
Ficheiros de texto

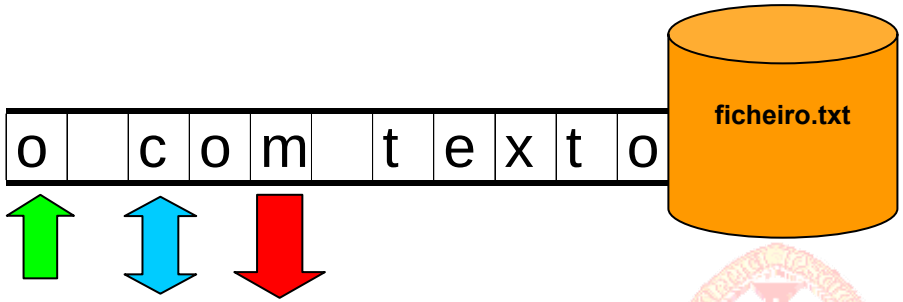
Streams de caracteres



Abertura de um ficheiro

- Tipo de dados FILE
  - Representa uma stream
- Só é possível declarar ponteiros
  - Os dados estão nos periféricos
  - O ponteiro contem o endereço do ficheiro para leitura/escrita
- Abrir um ficheiro (fopen)
  - Nome do ficheiro
  - Tipo de operações permitidas
    - Leitura
    - Escrita
    - Leitura/escrita





FILE * fopen(const char *filename, const char *mode)

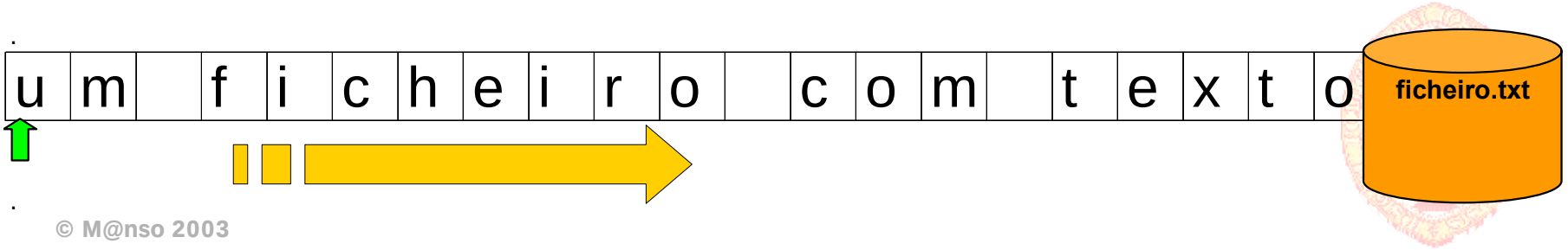
Modos de abertura do ficheiro

leitura ou escrita

- "r"
  - Abre o ficheiro para leitura
  - Se não puder abrir o ficheiro devolve NULL
- "w"
  - Abre o ficheiro para escrita
  - É criado um novo ficheiro destruindo o anterior caso exista.
  - Se não for possível abrir o ficheiro devolve NULL
- "a"
  - Abre o ficheiro para adição de dados (escrita)
  - Se o ficheiro existir o apontador aponta para o final do ficheiro
  - Se o ficheiro não existir é criado um novo ficheiro
  - Se não puder abrir o ficheiro devolve NULL

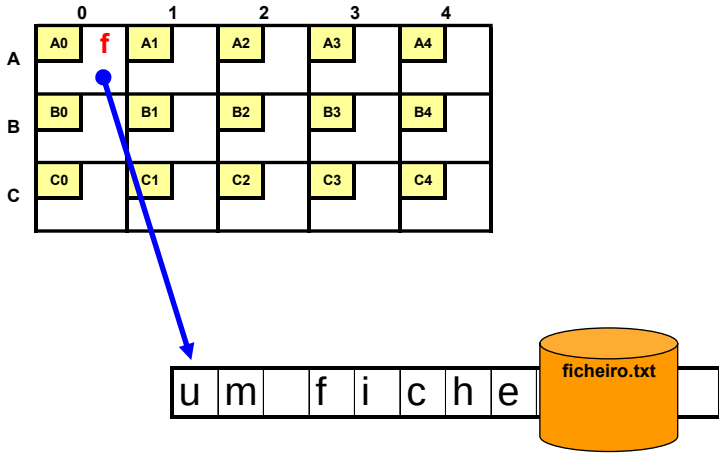
leitura e escrita

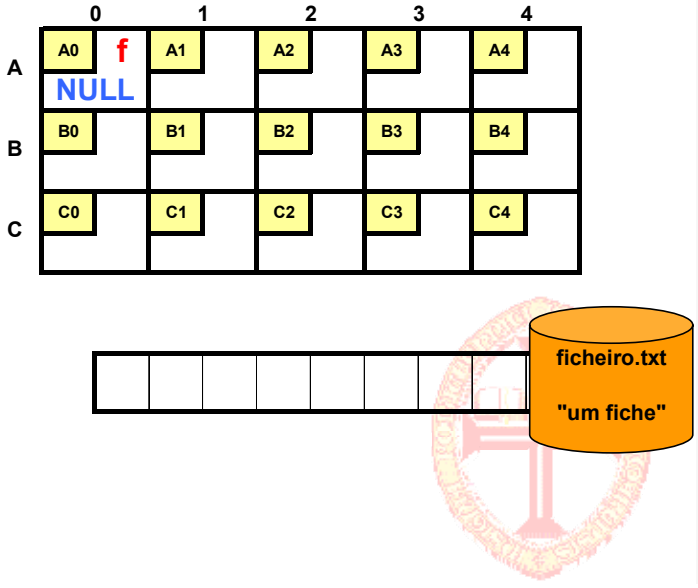
- "r+"
  - Abre o ficheiro para Leitura e Escrita
  - Se o ficheiro não existir cria um novo
  - Se existir posiciona-se no inicio do ficheiro
  - Se não puder abrir o ficheiro devolve NULL
- "w+"
  - Abre o ficheiro para Leitura e Escrita
  - Cria um novo ficheiro
  - Se não for possível abrir o ficheiro devolve NULL
- "a+"
  - Abre o ficheiro para Leitura e Escrita
  - Se o ficheiro não existir cria um novo
  - Se existir posiciona-se no fim do ficheiro
  - Se não puder abrir o ficheiro devolve NULL



Fechar um Ficheiro

- Fechar um ficheiro (fclose)
  - Grava fisicamente os dados
  - Retirar a ligação do ponteiro ao ficheiro
    - Libertar a memoria alocada pelo fopen.
  - int fclose(FILE * fich)
  - int fcloseAll()
- Esvaziar o buffer (flush)
  - Forçar os dados do buffer a serem escritos no ficheiro
  - int fflush(FILE * fich)
  - int fflushAll()



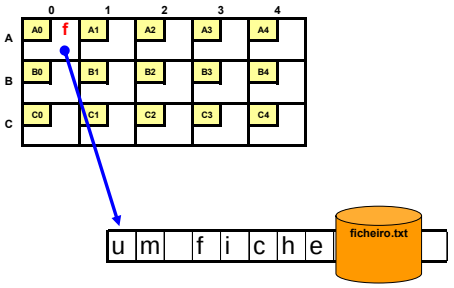


Exercício Abertura e Fecho de Ficheiros

- Construa um programa que abra um ficheiro para leitura cujo nome é introduzido pelo utilizador.

Abertura de um ficheiro

```
#include <stdio.h>
int main(int argc, char* argv[])
{
   FILE * f;
   char nome[100];
   printf("nome do ficheiro : ");
   scanf("%s",nome);
   f = fopen( nome , "r");
   if( f == NULL )
      printf("ERRO ao abrir o ficheiro");
   else{
      printf("Ficheiro Aberto");
      fclose(f);
   }
}
```

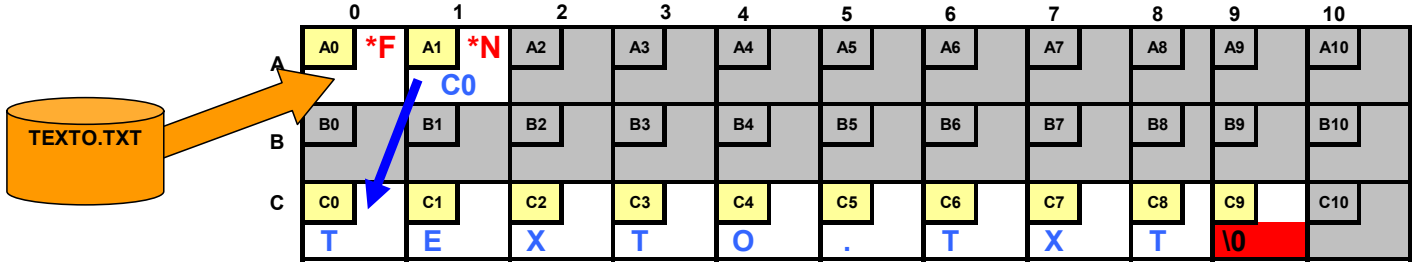


Exercício Abertura e Fecho de Ficheiros

- Construa uma função que verifique se um ficheiro existe. O nome do ficheiro deve ser passado como parâmetro.

Verifica se um ficheiro existe

```
int ExisteFicheiro(char *nomeFicheiro)
{
  FILE * f;
  f = fopen( nomeFicheiro , "r");
  if( f == NULL )
    return 0;
  else{
    fclose(f);
    return 1;
  }
}
```

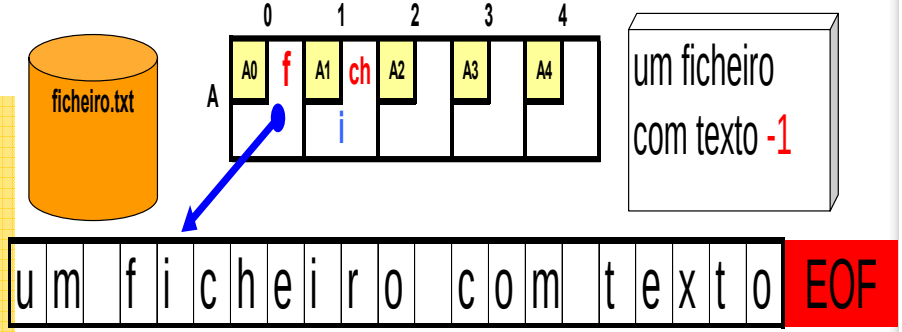


Leitura de caracteres

- int fgetc(FILE * f)
  - Recebe um ponteiro para um Ficheiro
  - Devolve
    - o valor ASCII do caracter lido
    - EOF se chegou ao fim do ficheiro
  - Avança o ponteiro para o próximo caracter

Leitura de um ficheiro de texto

```
FILE * f;
int ch;
f = fopen("ficheiro.txt","r");
do{
     ch = fgetc(f);
     printf("%c",ch);
} while( ch != EOF);
fclose(f);
```

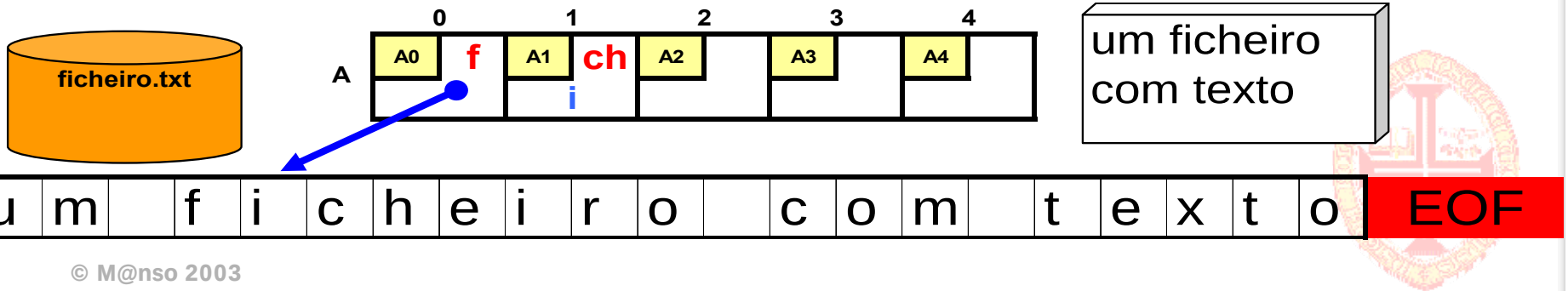


EOF – constante inteira que indica O fim do Ficheiro (-1)

Leitura de caracteres versão 2

Leitura de um ficheiro de texto

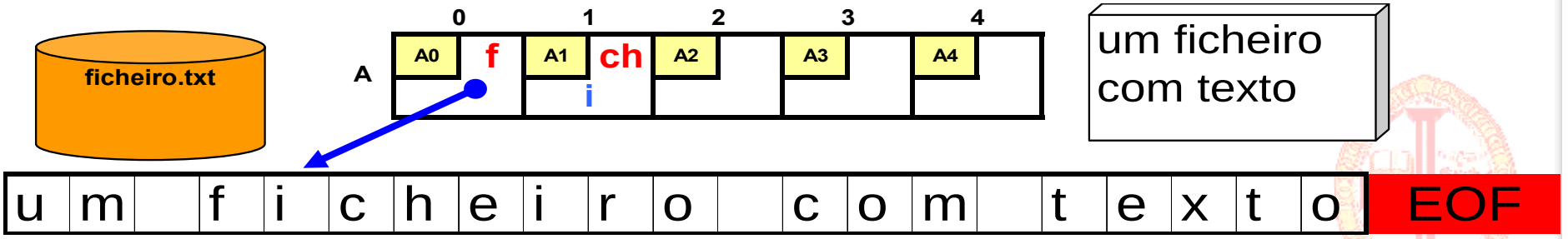
```
FILE * f;
int ch;
f = fopen( "ficheiro.txt" , "r");
if( f != NULL )
{
   do{
     ch = fgetc(f);
     if( ch != EOF)
        printf("%c",ch);
   } while( ch != EOF);
 fclose(f);
}
```



Leitura de caracteres versão 3

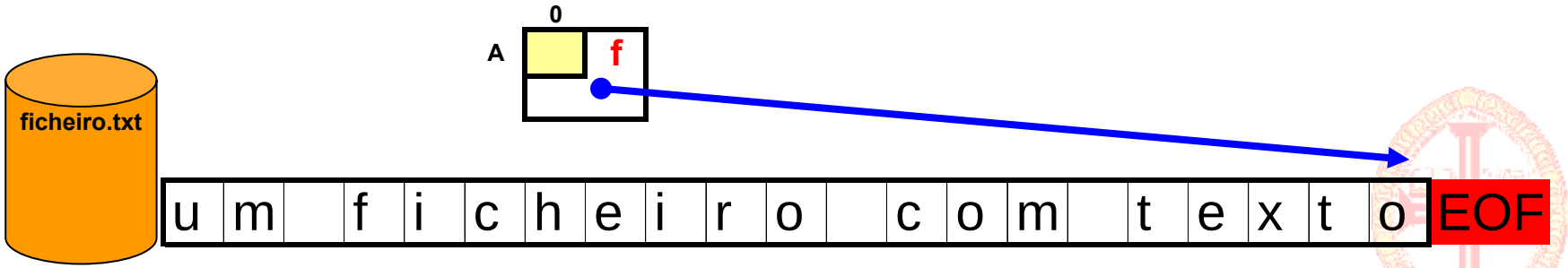
Leitura de um ficheiro de texto

```
FILE * f;
int ch;
f = fopen( "ficheiro.txt" , "r");
if( f != NULL )
{
   while( ( ch = fgetc(f) ) != EOF )
      printf("%c",ch);
 fclose(f);
}
```



escrita de caracteres

- Int putc (int ch , FILE * f)
  - Escreve um caracter no ficheiro
  - Recebe
    - O valor asccii do caracter a ser escrito
    - um ponteiro para um Ficheiro
  - Devolve
    - o caracter que escreveu
    - EOF se não consegui escrever
- int fflush(FILE *f)
  - Esvazia o buffer para o ficheiro
- Int fclose()
  - Acrescenta o EOF ao ficheiro (caso seja necessário)
  - Quebra a ligação do ponteiro

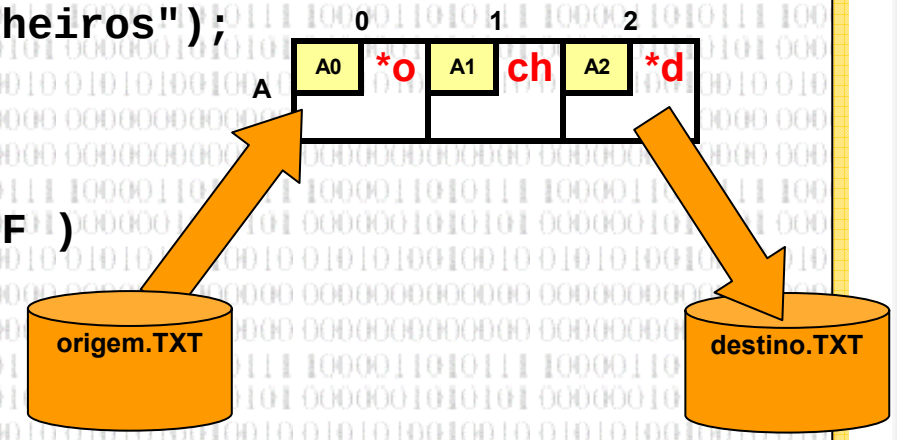


Exercício cópia de ficheiros

- Construa uma função que copie o conteúdo de um ficheiro para outro. O nome dos ficheiros devem ser passado como parâmetro.

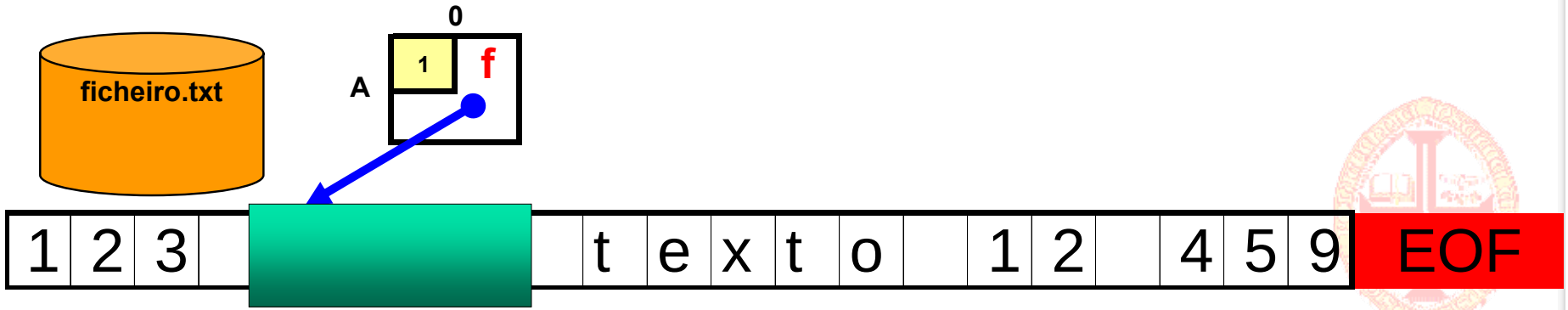
Copia um ficheiro

```
void copiaFicheiro(char *origem, char *destino){
  FILE *forig , *fdest;
  int ch;
  forig = fopen( origem , "r");
  fdest = fopen( destino , "w");
  if( forig == NULL  || fdest == NULL){
       printf("ERRO ao abrir os ficheiros");
       fcloseall();
       return;
  }
  while( (ch = fgetc(forig)) != EOF )
            fputc(ch,fdest);
  fclose(forig);
  fclose(fdest);
}
```



Leitura e escrita formatada

- int fprintf(FILE * f, char *formato)
  - Recebe
    - Um ponteiro para um ficheiro
    - O formato do dados a escrever
  - Devolve
    - o numero de caracteres que escreveu
    - EOF se não conseguir escrever
- int fprintf(FILE * f, char *formato)
  - Recebe
    - Um ponteiro para um ficheiro
    - O formato do dados a ler
  - Devolve
    - o numero de variáveis que leu
    - EOF se não conseguir ler

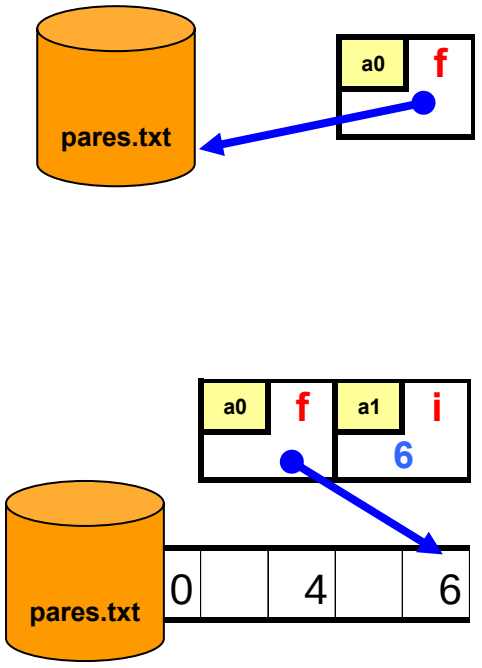


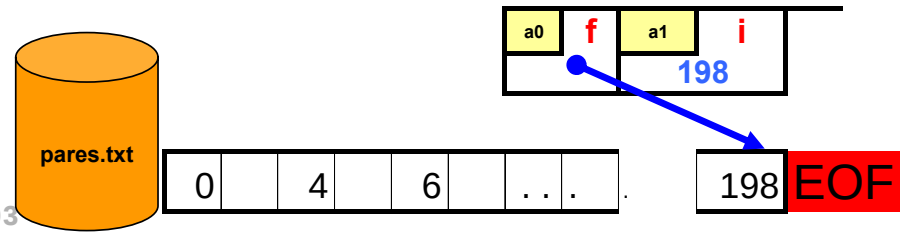
Exercício escrita formatada

- Construa um programa que escreva os primeiros 100 números pares no ficheiro “pares.txt”

Escrever números pares

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    int i;
    FILE *f = fopen("pares.txt","w");
    if( f != NULL ) {
        for( i= 0; i < 100 ; i++)
            fprintf( f, "%d ", i*2);
    fclose(f);
    }
}
```



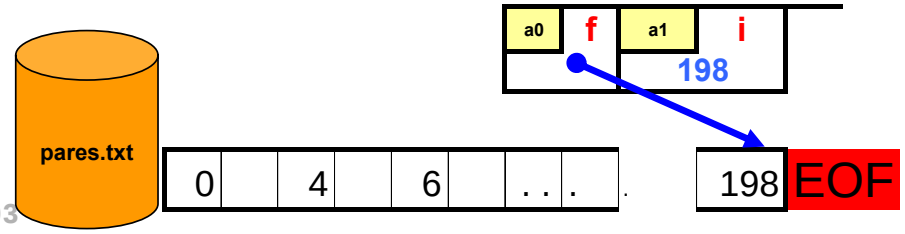


Exercício escrita formatada

- Construa um programa que leia os números inteiros que estão escritos no ficheiro “pares.txt”

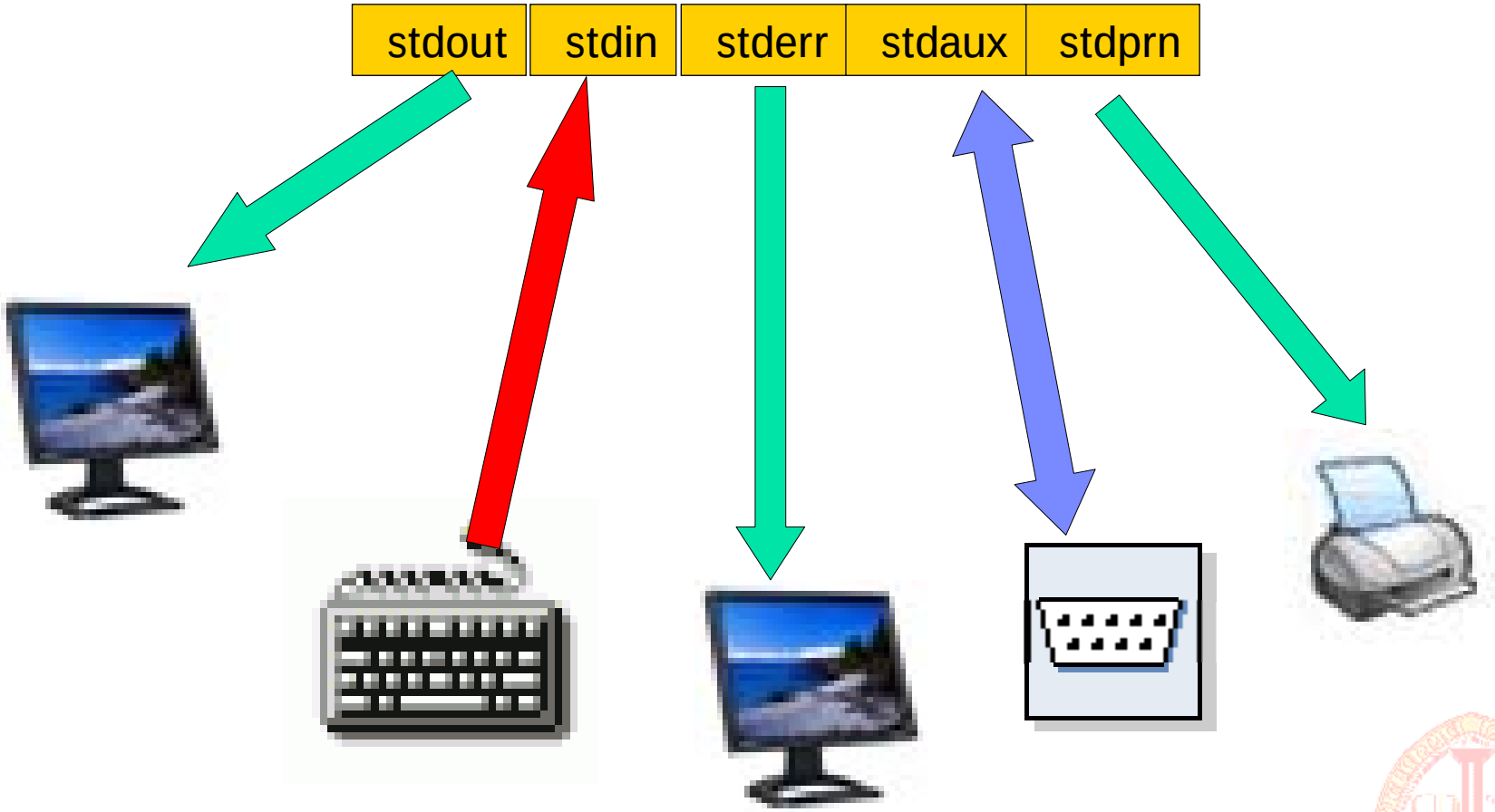
Ler números de um ficheiro

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    int i;
    FILE *f = fopen("pares.txt","r");
    if( f != NULL )  {
        while( fscanf(f,"%d",&i) != EOF)
            printf("%d ", i);
    fclose(f);
}
```



Ficheiros Standard

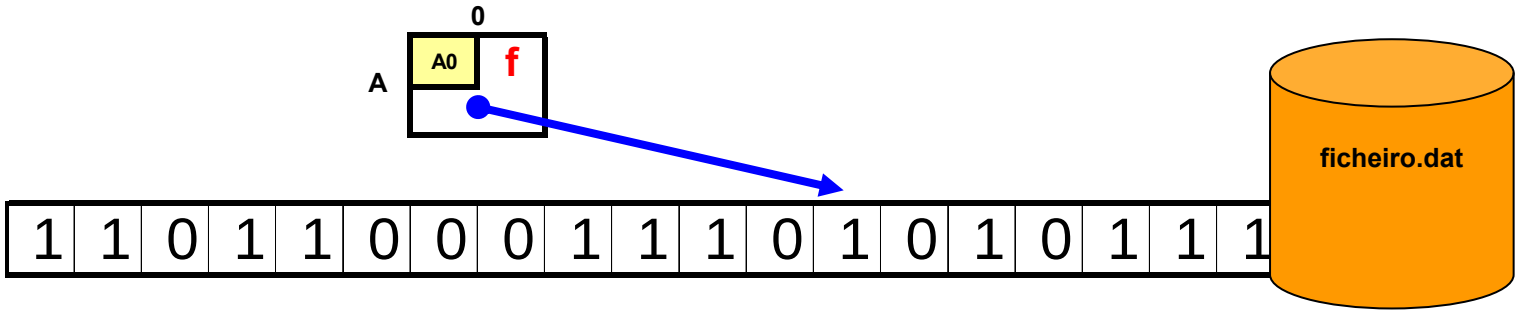
- #include <stdio.h>



printf(“um texto”);
fprintf(stdout, “um texto”);

Ficheiros binários

Ficheiros de bits



Ficheiros de texto e binários

Setup.ini

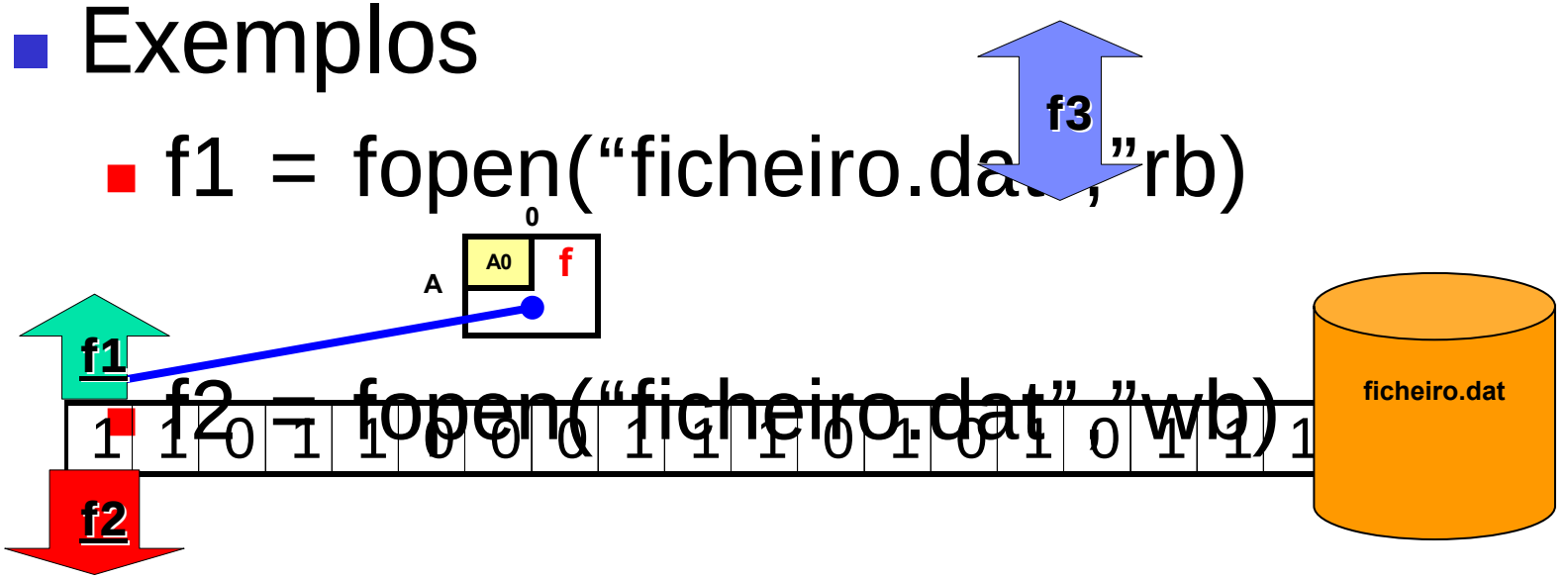
```
[Startup]
AppName=ATI Display Driver
FreeDiskSpace=0
[ISUPDATE]
UpdateURL=http://
[Languages]
Default=0x0009
count=24
key0=0x0009
key1=0x0011
key2=0x0804
key3=0x0404
key4=0x0005
key5=0x0006
```

Musica.mp3

```
¹Él ☺►
öà5º╝ ╔@ ┌è&¶ã0Ó!!kÂ¹=ÁXW
ù
©↕Ñ³p↔í#Bïâåƒ´;╬↑└¶µø1Kù
┬ÓX[f┌`:╠AêaÚ
°1♥►î<,Ó☻
♣▲—9ù@┐ ktK¤î♦DD▒nî'☼╨à
ëª»}Â▀ü╠Y½ ╠↔►╦x└Ç\f%Ã
¼↓A±?(_ëþ ←⌂@hïÉéd«ıSò0
Pñ+j{↔K♦9Õö┼]╙¥2èIåN³¬♂
8%Íb±-aGDáe´nH¯<&¥╕
¶êÿôÝÌò¢î²ãöeL±└¾u"»ì♥^═┴
å▄╝ ◄§uÑ5{ 7_
ÕWI}'í▀ pv▄mÂ▄má☻f═2
ÚÞ♣å╟Á↨2ÇÑxi▄Z{rÏÙ²KçÁþ§■
å_Ï⌂é½uØ&NÅNa▄0▒N³mÿAÅ
*¬/↨×
```

Ficheiros binários

- Abrir
  - fopen(char *nome, char *modo)
    - “rb” , “wb” , “ab”
    - “rb+” , “wb+” , “ab+”
- Exemplos
  - f1 = fopen(“ficheiro.dat”,”rb)
  - f2 = fopen(“ficheiro.dat”,”wb)
  - f3 = fopen(“ficheiro.dat”,”wb+)

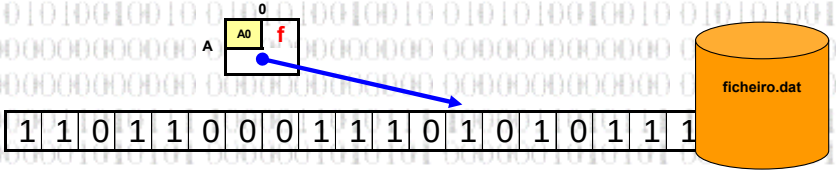


Ficheiros binários

- Escrever
  - Int fwrite(void *ptr, int tamDados, int numDados, FILE *ficheiro)
    - parametros
      - ptr – local da memória onde estão os dados
      - tamDados – numero de bytes que cada elemento contem
      - numDados – numero de dados a escrever
      - icheiro – stream do ficheiro de escrita
    - Retorna o número de bytes que escreveu
- Exemplo

Escrever um número num ficheiro binário

```
int num = 10;
FILE *file;
file = fopen("dados.dat","wb");
fwrite( &num, sizeof(num), 1,file)
. . .
fclose(file);
```

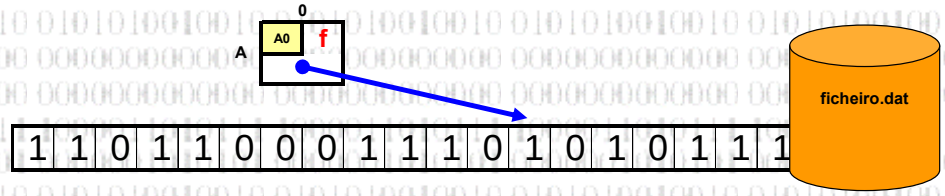


Ficheiros binários

- Ler
  - int fread(void *ptr, int tamDados, int numDados, FILE *ficheiro)
    - parametros
      - ptr – local da memória onde estão os dados
      - tamDados – numero de bytes que cada elemento contem
      - numDados – numero de dados a escrever
      - icheiro – stream do ficheiro de escrita
    - Retorna o número de bytes que escreveu
- Exemplo

Ler um número de um ficheiro

```
int num;
FILE *file = fopen("dados.dat","rb");
fread( &num, sizeof(num) , 1, file);
. . .
fclose(file);
```



Exercício

- Escreva um programa que escreva 100 números aleatórios entre 1 e 200 num ficheiro binário chamado “dados.dat”

Escrever números num ficheiro binário

```
int i, num;
   FILE *file;
   file = fopen("dados.dat","wb");
   if( file == NULL){
      printf("não é possivel abrir o ficheiro");
      return;
   }
   randomize();
   for( i=0 ; i < 100 ; i++ ){
      num = random(199) + 1;
      fwrite( &num, sizeof(num), 1,file);
   }
   fclose(file);
```

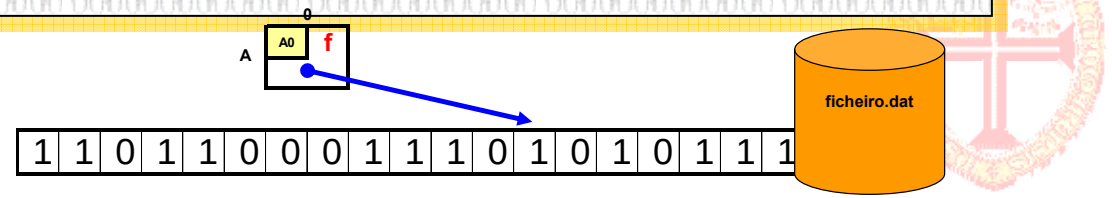
Exercício

- Escreva um programa que leia números num ficheiro binário chamado “dados.dat” e os apresente no ecran

Ler números de um ficheiro

```
int i, num;
  FILE *file;
  file = fopen("dados.dat","rb");
  if( file == NULL){
     printf("não é possivel abrir o ficheiro");
     return;
  }
 while(  fread( &num, sizeof(num) , 1, file) > 0 )
     printf("\t%d",num);

  fclose(file);
```



Exercício

- Escreva um programa que leia 100 números aleatórios num ficheiro binário chamado “dados.dat” e os apresente no ecran

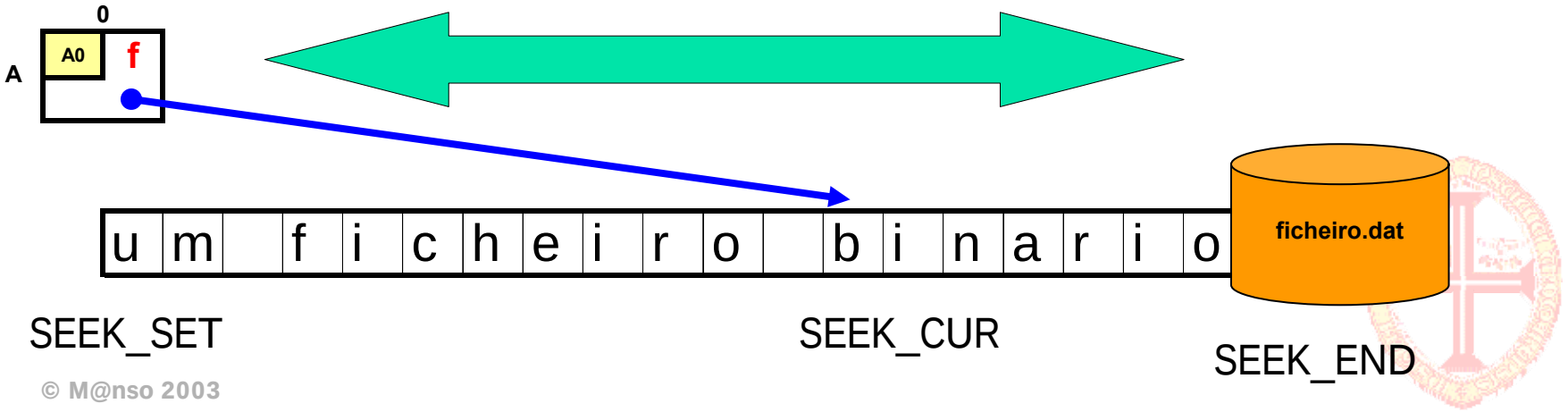
Ler números de um ficheiro

```
int i, num;
FILE *file;
int data[100];
file = fopen("dados.dat","rb");
if( file == NULL){
   printf("não é possivel abrir o ficheiro");
   return;
}
fread( data, sizeof(num) , 100, file);
for( i=0 ; i < 100 ; i++ )
   printf("\t%d",data[i]);

fclose(file);
```

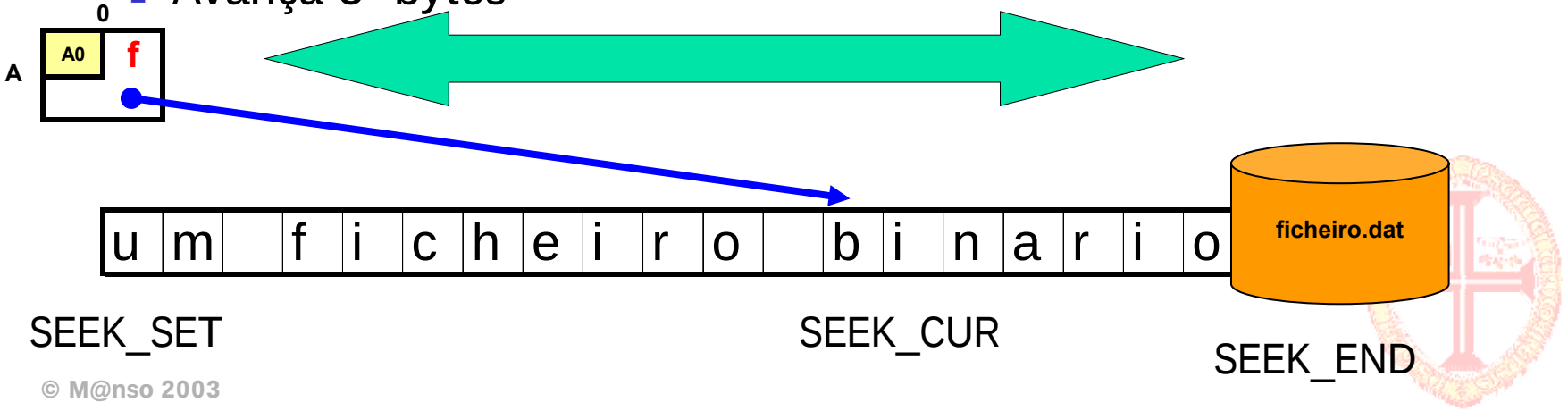
Acesso directo á Stream

- Acesso directo á stream
  - fseek(FILE *ficheiro, long salto , int origem)
    - parâmetros
      - ficheiro – stream do ficheiro
      - Salto - Numero de bytes do salto
      - Origem do salto
        - SEEK_SET – inicio da stream
        - SEEK_CUR – posição corrente
        - SEEK_END – fim da stream
  - As leituras e escritas são feitas no local onde o ponteiro está
  - Sempre que se faz uma leitura ou uma escrita o ponteiro avança



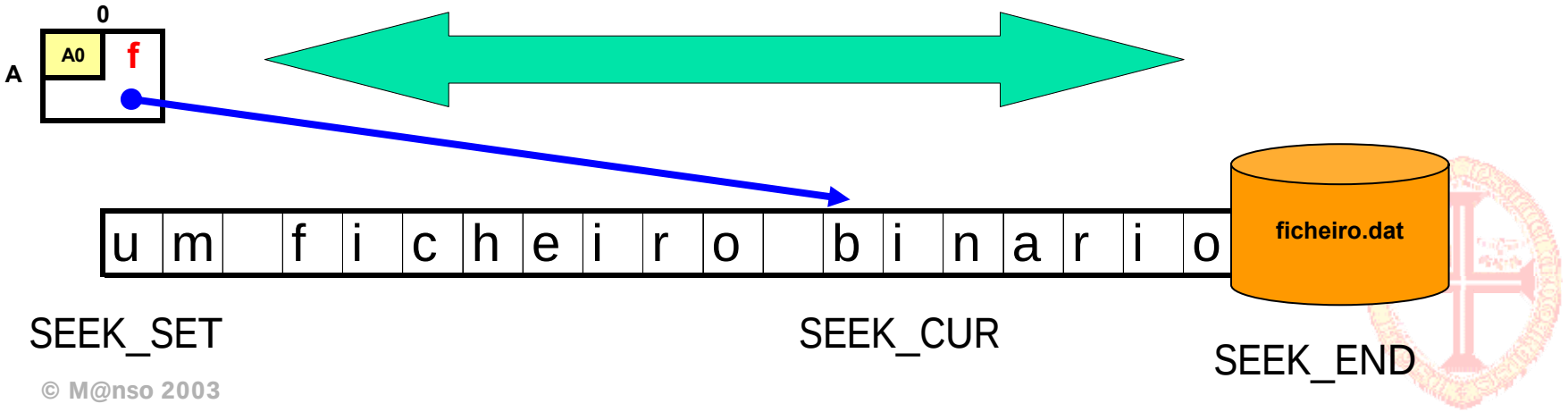
Acesso directo á Stream

- Exemplos
  - fseek(f,0, SEEK_SET)
    - Vai para o inicio
  - fseek(f,0, SEEK_END)
    - Vai para o fim
  - fseek(f,-3, SEEK_CUR)
    - Recua 3 bytes
  - fseek(f,5, SEEK_CUR)
    - Avança 5 bytes



Acesso directo á Stream

- Acesso directo á stream
  - ftell(FILE *f)
    - Devolve a posição (em bytes) onde está o ponteiro f
  - rewind(FILE *f)
    - Coloca o ponteiro no inicio
    - fseek(f,0, SEEK_SET)

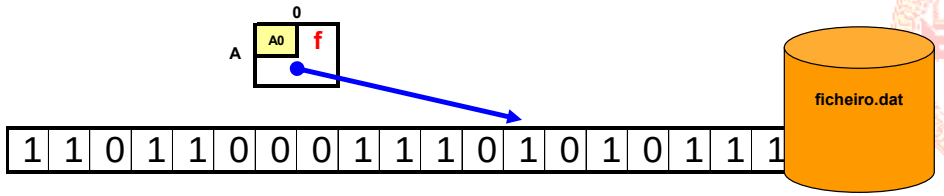


Exercício

- Escreva uma função que calcule o tamanho de um ficheiro cujo nome é passado por parâmetro

Escrever números num ficheiro binário

```
int bytesFicheiro(char *nome){
    FILE *file;
    file = fopen(nome,"rb");
    if( file == NULL){
        printf("não é possivel abrir o ficheiro");
        return 0;
    }
    fseek(file,0,SEEK_END);
    return ftell(file);
}
```



Exercício

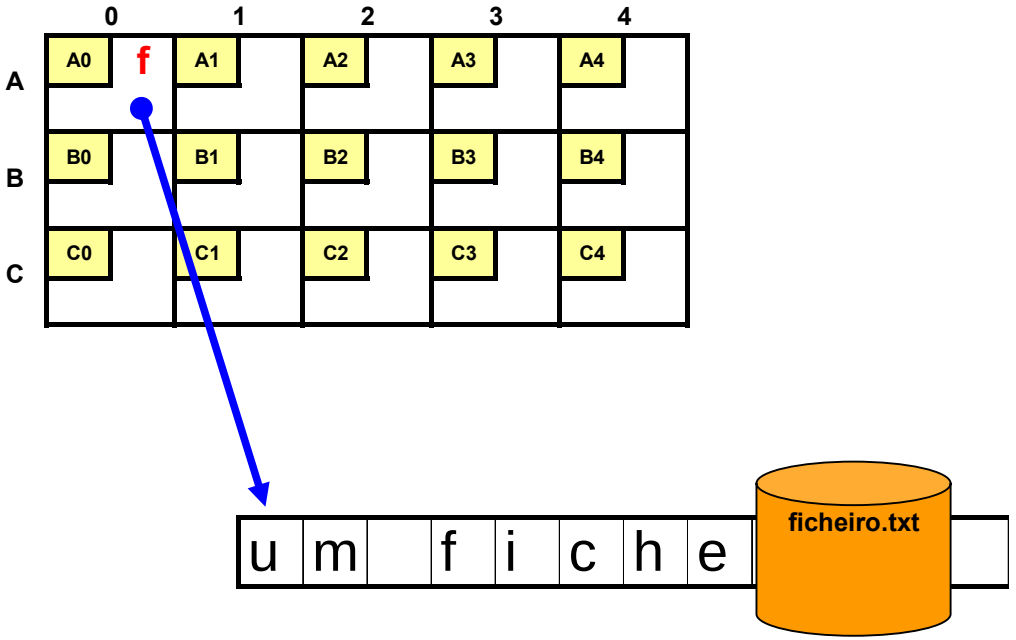
- Escreva uma função que copie um ficheiro binário. O nome do ficheiro de origem e destino devem ser passados como parâmetro.

Copiar um ficheiro binário

```
void copiaBinario(char *origem, char *destino){
   FILE *forig,  *fdest;
   forig = fopen(origem,"rb");
   fdest = fopen(destino,"wb");
   if( forig == NULL || fdest == NULL){
      printf("não é possivel abrir os ficheiros");
      fcloseall();
      return ;
   }
   char byte;
   while( fread(&byte, sizeof(byte),1,forig) > 0)
     fwrite(&byte, sizeof(byte),1,fdest);
   fcloseall();
}
```

Resumo

- Ficheiros são ponteiros para streams
- Streams são conjuntos de dados sequenciais que residem em periféricos.
- Utilização de streams
  - Abrir a stream – fopen
    - Texto / binário
    - Leitura / escrita
  - Utilizar a stream
    - Ler
      - fget , fscanf , fread
    - Escrever
      - fputc, fprinf , fwrite
    - Movimentar
      - fseek
  - Fechar a stream
    - Esvaziar o buffer - fflush
    - Fechar a stream - close



Dúvidas