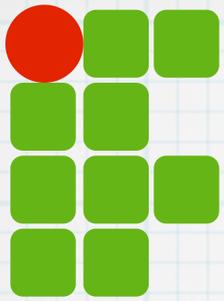
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Algoritmos

ANSI C - Arquivos



Agenda

- Conceito
- Tipos de arquivos
  - Texto
  - Binário
- Stream
- Principais funções
- Exemplos

Arquivo

* Um arquivo é uma seqüência de bytes localizada em um dispositivo de armazenamento
  * Disco rígido (HDD), CD-ROM, Pen-Drive, Cartão SD, etc
* Programas de computador podem ler e escrever dados em arquivos
  * A interpretação semântica dos bytes é de responsabilidade do programa

Tipos de arquivo

* Caracteres (texto)
  * Cada byte é um caractere (ASCII)

H	e	l	l	o		w	o
r	l	d	\n	EOF			

* Binários
  * Inteiros, real, lógico, etc.

0A	A1	51	65	92	11	32	12
68	F1	32	12	AB	A0	B5	C4
B9	EOF						

EOF = End Of File
Código ASCII 04

Tipos de arquivo

* Considere um arquivo que armazena dois número inteiros: 123456 e 987654
  * O arquivo binários usará 8 bytes, considerando um inteiro com 32 bits (4 bytes)
  * O arquivo texto usará 13 bytes, pelo menos, um para cada caracter e um caracter que separa os dois números

BINÁRIO

000	000	111	010
000	000	000	000
EOF			

TEXTO

1	2	3	4	5	6	\n	9
8	7	6	5	4	EOF		

Arquivo texto

arq1.txt

```
Este arquivo contém apenas texto!
```

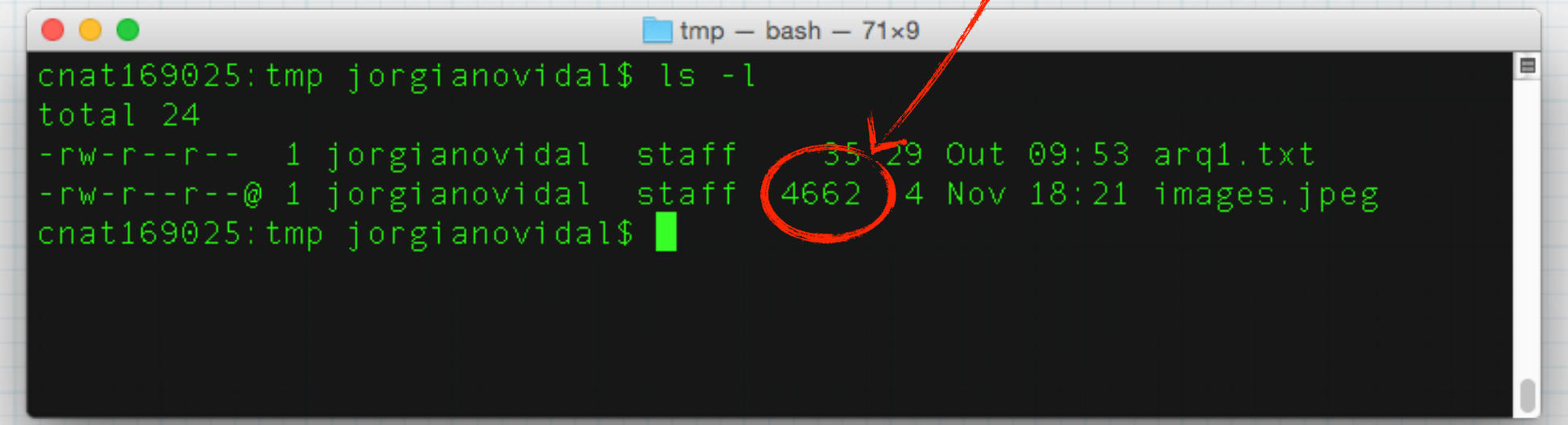
Seqüência com 35 bytes. Além dos 33 bytes que forma o texto tem um byte pra a quebra de linha e um para o EOF

tmp — bash — 64×11

```
ead113925:tmp jorgianovidal$ ls -l
total 8
-rw-r--r--  1 jorgianovidal  staff  35 29 Out 09:53 arq1.txt
ead113925:tmp jorgianovidal$
```

Arquivo binário

Sequencia de 4662 bytes que contem as informações da imagem



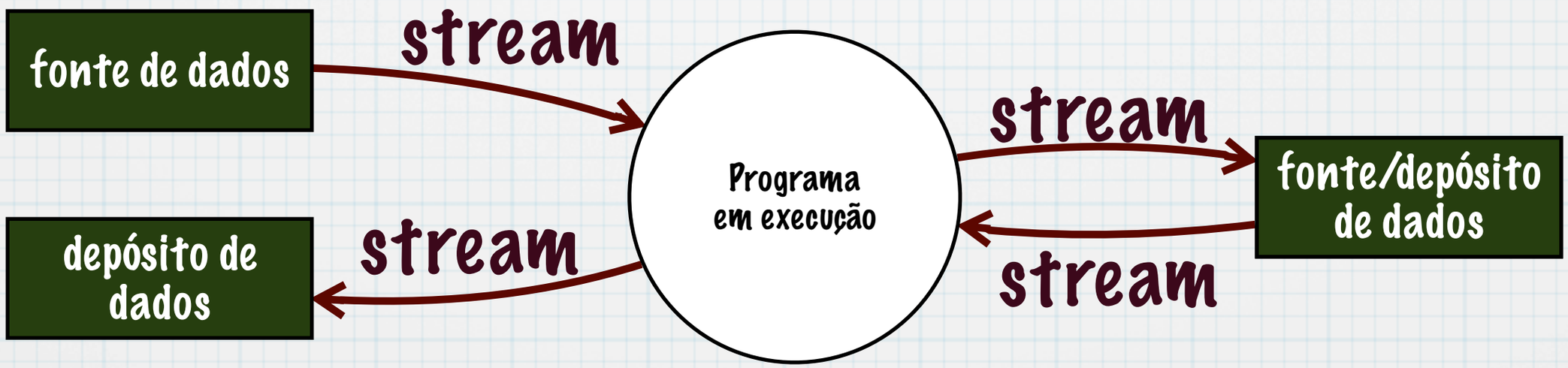
Arquivo binário

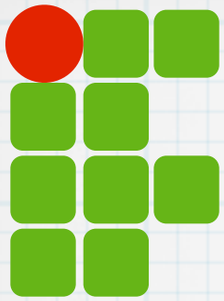
images.jpeg

```
ˇÿˇ‡JFIFˇ€Ñ     &$/ $'),,,150*5&+,)


,$,,,,,,,,,,,,,,,,,,,,,,,),,,,,,,,,,,,,,,),,,,,),),ˇ¿¬"ˇƒˇƒ>!1AQaqÅë°±—◆"ì·2BíÒRr
¢Sb≤Ç“#ˇƒˇƒ&!AQ1Ba°±q2ˇ/
?ˇxDlbHDtQfléTb6(ëC†4*Dl >0Adã°>—63Aèä$i°±¶%ßò§`6 ämRb®¿lb\PƒÑ¥“$bÄIÑÖŸ¥ë¶:1
à¥tdb>:Íí™]»ô&uÎ2êH ªñäA 0êIÇ∞êh√ã∞”X^±WGÿÓ@nCh6·ëQ∞Âù±B¢1
¯âû¢Õ¶>#¢ĀÒXmù
ãëFî¯»tû#¢-áŸ C"≈DdD1±
±B”      ,
ÅDQ
ßÜ)ò!Dh$
-†—H
 ê(¥-aÅL¥(
Â‹íÊa\Ö¿,C"âéå§Øï6(∏”4«ï¥(!—«Fì6√Upi±—¶
aÙ8çä4ÿ»@]ûJ8°—EBì©πCÒ©çä*0
b/ié“1óccLù )0¥à≈±¶â;ö≥§RÄj#R   !yèΔRA$5Dñ ‡ÇHÚ¥Ôj¥j2¥Í.-bïïÛ[rÌ>K•}È§/°´≤SNÔé≠Ìn”Y1◆«Ï8∫
í~Ò4óïKrÑ<±G˜è•-™[qåm Î"?¨√’m_N≤ëv9ŒÖÔRJ ≠5-
Ó7O»›Gfi≈&Ò§“Ÿi_8f4 ûùÚπA#,ßÔSGI˜ÕÀ˙Sè ªvl<flı¶öxËÓ‹⁎&˘Í4Í·~´:A,r˝#flVZö<W˜Mø˙•c:˜≈Y›¸:8coü-
flò◊´Ñ°øËj˙uãîrïÔ™_Í”Ìô
ÚcÔ˙°fl”œÇ&‡Œªƒ}8éåER4¡åÓm⁎C„)
D^∑a$:ÙÅOç`60⁎#b%ßö4hPB†Δ)ªZXli†“B£!ââ™iî2(4.,/â›ü ,<å—&õÌ~çI§Í)6Ï‘>}^2iŸx◆>{¶˝ÂŸ[E¶Â-
≥ö˘V-Y$Ï◆≥Ω^/€ßóS
~Ër}‰™%k¥ÆÏÆ“ªy%}º'§˝Ø—¥yjTõr≤z∞ÑÊÏÚΔ⁎ÀY‰›#”ZFëSZΩUÜJÍÀîpäÃœ˘…Δ?,oœc{mgc…üÂ_Ÿé
€ªÎ˙c/àÈîÔ^0Ö‘uÈ¶üÍû+≥?N{yRPt© nú†¢‹‘Tfi2÷NÌlæn¯ü)¯Õl\Ø√+u
≠—ØəÁüÂœ+wM&·R”[x´ºWoÙÈ ¶[2±äM¥◊î7
˘Âµ!˙E7™ıØªn6eRåõΩÏ∫º¬≠“¢IŸÍ‰^€w!
ªlFíÎªqç≥ñ≠o>°t…7l$Æ/◆ñµ€m§’û
m›vjß™°t¢ö√Ô/ÒxœÂß{°h/cI)%&≤m&◊Y¶=)”Ì˘QäzE∏^¥´e‡N·2ª—æõø—
˘lrÏV≤wVJ¸∫“1≠>[˙ÆJöT^2X≠â·÷oè^ÿ©¬I·N
o∑‹Ç_LIe‡ºã)¨“•}§G@œ    çåè©éCu&iÅÇïCT⁎≥lE!\|+
a/"á@D⁎çT⁎≤ù®Tj!±íû

ä6m0ëÊÙø¥‘4d¸$÷µø"ΔMÏVYuÿÚ}·t‹¥}?n©=T”iÍ≈7+5ñ:´¨‰Íəºæi<qrwoû^œ?S©q˙<õ}æùÔ•Lû
¢≥ioÿ∏ûVì”ïka9fi+{§≠vöIcõÿ|ÁƒΔÀ"•_âœœ
≤ÛTfl◆^÷õl∞Ÿπc{˝33i:R[∫ëÊ:¯`
Ó≥':ÔGπı´flfk≥ñ,Sjœ÷„<Ôºª∏ÏØi™—Rə6]æ[Ö mÊ —·≠++æX≥']§öΩù∞…,Øã|âÂî¿¯„ñ_OWaSä€ôJv||
Q·ûe(jè7dÜ:v’rN€«ËÚJ…cï¸£tãF<∑≠Áü,Ó1£qfu^Q≈•Ö◆œcfim!π∏‰∞ø;b°ÿΩ.≠öQx‰¯YÿFµπ±Ò√…d”eZä÷œ÷·.
```

Stream

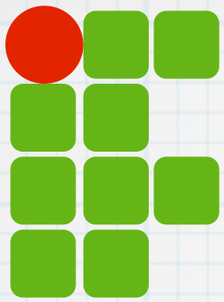
* Para ter acesso aos bytes de um arquivo, normalmente usamos streams
* Uma Stream (em programação C) é uma seqüência de dados disponível para processamento





Stream

* Acesso a arquivos
  * Seqüencial
    * Os bytes são lidos/escritos 1 por vez e não há retorno
    * exemplo: monitor e teclado
  * Aleatório
    * Podemos avançar e retroceder
    * Um indicador de posição é associado a stream de acesso ao arquivo
    * exemplo: arquivos de programas salvos no HDD



ANSI C

* A biblioteca de C fornece uma API para acessar um arquivo qualquer através de uma stream
* O acesso é feito através de um ponteiro para arquivo, definido em <stdio.h>
* Declaração do ponteiro:

```
FILE * arquivo;
```

Principais funções

fopen	Abre um arquivo
fclose	Fecha um arquivo
fread	Ler dados de um arquivo
fwrite	Escreve dados em um arquivo
rewind	Volto o indicador de posição para o início do arquivo
fseek	Deslocar o indicador para uma posição específica
fflush	Salva os dados do buffer no arquivo alvo
fprintf	Escreve dados em um arquivo
fputs	Escreve string em um arquivo
fscanf	Ler dados de um arquivo
fgets	Ler string de um arquivo
ferror	Retorna verdadeiro se houve erro em algum processamento
feof	Retorna verdadeiro se o indicador de posição estiver no fim do arquivo

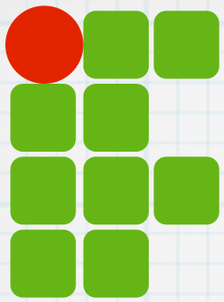
Para abrir um arquivo

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

filename	string com o nome do arquivo
modo	Mode de abertura do arquivo

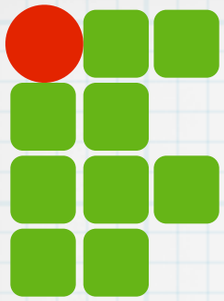
Modos de abertura

r	Abrir para leitura
w	Abrir para escrita
a	Abrir para adicionar (append)
r+	Abrir para leitura/escrita. A stream apontará para o início do arquivo
w+	Abrir para leitura/escrita. Será criado um arquivo se o mesmo não existir
a+	Abrir para leitura/escrita. A stream apontará para o final do arquivo
rb	Abrir um arquivo binário (b) para leitura
wb	Abrir um arquivo binário (b) para escrita
ab	Abrir um arquivo binário (b) para adicionar



Exemplos

```c
FILE  *f1, *f2, *f3;

/*Abre "stats.fft" Para escrita binária */
f1 = fopen("stats.fft", "wb");

/* Abre "nomes.txt" para leitura*/
f2 = fopen("nomes.txt", "r");

/* Abre "agenda.dat" para leitura e escrita  */
f3 = fopen("agenda.dat","r+");
```



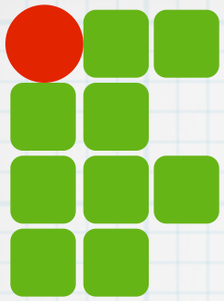
Escrever dados em um arquivo

* Arquivo texto
  * fprintf

```
int fprintf(FILE * arquivo,  char * formato, ...);
```

* fputs

```
int fputs( char * s, FILE * arquivo);
```



Exemplo

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv) {
    FILE *agenda;
    char nome[80], telefone[20];
    agenda = fopen("agenda.txt", "w");
    fputs("*** Agenda de Contatos ***\n", agenda);
    do {
        printf("Nome:");
        gets(nome);
        printf("telefone: ");
        gets(telefone);
        if (strcmp(nome, "fim") != 0) {
            fprintf(agenda, "%s:%s\n", nome, telefone);
        }
    } while (strcmp(nome, "fim") != 0);
    fclose(agenda);
    return 0;
}
```

Exemplo

Abre o arquivo

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv) {
    FILE *agenda;
    char nome[80], telefone[20];
    agenda = fopen("agenda.txt", "w");
    fputs("*** Agenda de Contatos ***\n", agenda);
    do {
        printf("Nome:");
        gets(nome);
        printf("telefone: ");
        gets(telefone);
        if (strcmp(nome, "fim") != 0) {
            fprintf(agenda, "%s:%s\n", nome, telefone);
        }
    } while (strcmp(nome, "fim") != 0);
    fclose(agenda);
    return 0;
}
```

Exemplo

Abre o arquivo

Escreve um cabeçalho

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv) {
    FILE *agenda;
    char nome[80], telefone[20];
    agenda = fopen("agenda.txt", "w");
    fputs("*** Agenda de Contatos ***\n", agenda);
    do {
        printf("Nome:");
        gets(nome);
        printf("telefone: ");
        gets(telefone);
        if (strcmp(nome, "fim") != 0) {
            fprintf(agenda, "%s:%s\n", nome, telefone);
        }
    } while (strcmp(nome, "fim") != 0);
    fclose(agenda);
    return 0;
}
```

Exemplo

Abre o arquivo

Escreve um cabeçalho

Escreve os dados

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv) {
    FILE *agenda;
    char nome[80], telefone[20];
    agenda = fopen("agenda.txt", "w");
    fputs("*** Agenda de Contatos ***\n", agenda);
    do {
        printf("Nome:");
        gets(nome);
        printf("telefone: ");
        gets(telefone);
        if (strcmp(nome, "fim") != 0) {
            fprintf(agenda, "%s:%s\n", nome, telefone);
        }
    } while (strcmp(nome, "fim") != 0);
    fclose(agenda);
    return 0;
}
```

Exemplo

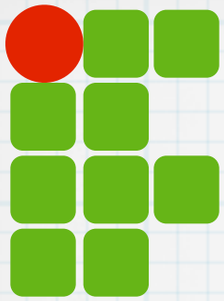
Abre o arquivo

Escreve um cabeçalho

Escreve os dados

Fecha o arquivo

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv) {
    FILE *agenda;
    char nome[80], telefone[20];
    agenda = fopen("agenda.txt", "w");
    fputs("*** Agenda de Contatos ***\n", agenda);
    do {
        printf("Nome:");
        gets(nome);
        printf("telefone: ");
        gets(telefone);
        if (strcmp(nome, "fim") != 0) {
            fprintf(agenda, "%s:%s\n", nome, telefone);
        }
    } while (strcmp(nome, "fim") != 0);
    fclose(agenda);
    return 0;
}
```



```c
int main(int argc, char **argv) {
    FILE *agenda;
    char cabecalho[80], linha[100];
    agenda = fopen("agenda.txt", "r");
     fgets(cabecalho,80,agenda);
     if (!(strcmp("*** Agenda de Contatos ***\n",cabecalho)==0)){
        printf("Erro: Arquivo não é do tipo agenda de contatos.");
        exit(1);
     }
     while (!feof(agenda)){
        fgets(linha,100,agenda);
        printf("%s",linha);
     }
     fclose(agenda);
    return 0;
}
```

Dicas

* Sempre verifique o sucesso ou fracasso de uma operação com arquivos
* Exemplo: Verificar se abertura ocorreu corretamente

```c
agenda = fopen("agenda.txt", "w");
if (agenda==NULL){
    printf("Erro ao abrir agenda de contatos\n");
    exit(1);
}
```

Arquivos binários

* Ao abrir, informamos o tipo

```
FILE * numeros;
numeros = fopen("numeros.dat","wb");
```

Binário

* Principal comando de escrita

```
size_t fwrite(void * ptr, size_t size, size_t nitems, FILE * stream);
```

Arquivos binários

* Ao abrir, informamos o tipo

```
FILE * numeros;
numeros = fopen("numeros.dat","wb");
```

Binário

* Principal comando de escrita

```
size_t fwrite(void * ptr, size_t size, size_t nitems, FILE * stream);
```

Dados a serem escritos

Arquivos binários

* Ao abrir, informamos o tipo

```
FILE * numeros;
numeros = fopen("numeros.dat","wb");
```

Binário

* Principal comando de escrita

```
size_t fwrite(void * ptr, size_t size, size_t nitems, FILE * stream);
```

Dados a serem escritos

Tamanho do dado em bytes

Arquivos binários

* Ao abrir, informamos o tipo

```
FILE * numeros;
numeros = fopen("numeros.dat","wb");
```

Binário

* Principal comando de escrita

```
size_t fwrite(void * ptr, size_t size, size_t nitems, FILE * stream);
```

Dados a serem escritos

Tamanho do dado em bytes

Quantidade de dados (array)

Arquivos binários

* Ao abrir, informamos o tipo

```
FILE * numeros;
numeros = fopen("numeros.dat","wb");
```

Binário

* Principal comando de escrita

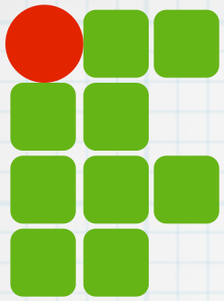
```
size_t fwrite(void * ptr, size_t size, size_t nitems, FILE * stream);
```

Dados a serem escritos

Tamanho do dado em bytes

Quantidade de dados (array)

Arquivo



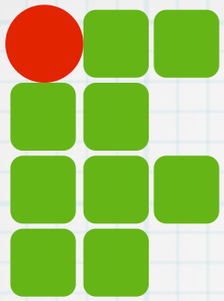
Escrita

```c
int main(int argc, char **argv) {
   int i;
   int quantidade, nums[20];
   FILE * numeros;
   quantidade = 5;
   for (i = 0 ; i < quantidade ; i++)
      scanf("%d",&nums[i]);
   numeros = fopen("numeros.dat", "wb");
   fwrite(&quantidade,sizeof(int),1,numeros);
   fwrite(&nums,sizeof(int),quantidade,numeros);
   fclose(numeros);
   return 0;
}
```

Escrita

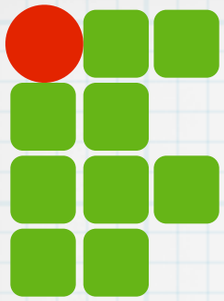
```c
int main(int argc, char **argv) {
   int i;
   int quantidade, nums[20];
   FILE * numeros;
   quantidade = 5;
   for (i = 0 ; i < quantidade ; i+
      scanf("%d",&nums[i]);
   numeros = fopen("numeros.da",  "wb");
   fwrite(&quantidade,sizeof(int),1,numeros);
   fwrite(&nums,sizeof(int),quantidade,numeros);
   fclose(numeros);
   return 0;
}
```

A função sizeof informa a quantidade de bytes de um determinado tipo



A função sizeof

* Informa o tamanho de um tipo de dado
  * O `int` pode ter tamanhos diferentes, por exemplo
* SEMPRE use sizeof para saber o tamanho do tipo
  * Isto torna o programa mais portável
* `sizeof(tipo)` retorna o tamanho, em bytes, de tipo
  * Exemplo: `sizeof(int)` retorna 4 para arquitetura de 32 bits.

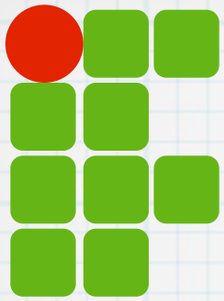


Leitura

* Usamos a função fread

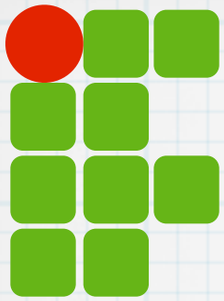
```
size_t fread(void * ptr, size_t size, size_t nitems, FILE * stream);
```

* É necessário saber o tamanho e a quantidade de dados a serem lidos
  * Conhecimento detalhado do conteúdo do arquivo



Exemplo

```c
int main(int argc, char **argv) {
   int i;
   int quantidade, nums[20];
   FILE * numeros;
   numeros = fopen("numeros.dat", "rb");
   fread(&quantidade,sizeof(int),1,numeros);
   fread(nums,sizeof(int),quantidade,numeros);
   fclose(numeros);
    for (i = 0 ; i < quantidade ; i++){
     printf("NUM[%d] = %d\n",i,nums[i]);
    }
   return 0;
}
```



Cópia de arquivo

✓

```c
int main(int argc, char **argv) {
    FILE * origem, *destino;
    unsigned char buffer[101];
    int quantidade;
    /* Verifica os parametros */
    origem = fopen(argv[1], "rb");
    destino = fopen(argv[2], "wb");
    quantidade = fread(buffer, sizeof(unsigned char), 100, origem);
    while (quantidade > 0) {
        fwrite(buffer, sizeof(unsigned char), quantidade, destino);
        quantidade = fread(buffer, sizeof(unsigned char), 100, origem);
    }
    fclose(origem);
    fclose(destino);
    return 0;
}
```

Dúvidas?