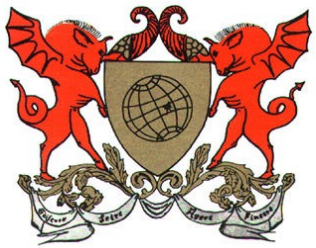


Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

Aula de Hoje

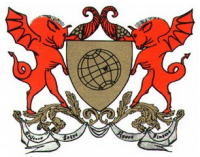
Arquivos

Arquivos

O computador basicamente trabalha com dois ambientes de armazenamento:

A memória primária (RAM) que é mais rápida porém extremamente volátil.

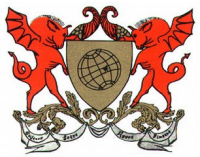
E as memórias secundárias (HD, CD, DVD, PenDrive, etc.) que são menos voláteis.



Arquivos

Existem aplicações que necessitam de armazenamento não-volátil e para isso existem os chamados arquivos.

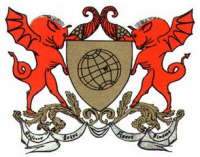
Arquivos são coleções de bytes que são armazenados em dispositivos de armazenamento secundário e são referenciados por um nome único.



Arquivos

Não há perda de dados ao se desligar o computador e/ou o programa. O que não acontece com dados em variáveis na memória principal.

Os arquivos podem ser lidos e alterados.



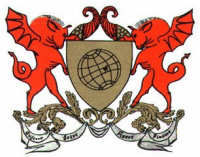
Arquivos

Vantagens de se usar arquivo:

-Armazenamento durável;

-Permitem armazenar **uma grande quantidade** de informação;

-Acesso concorrente aos dados;



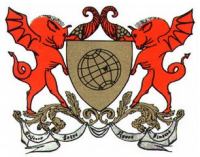
Arquivos

Cuidado:

A extensão do arquivo não define o seu tipo.

O que define um arquivo é a maneira como os dados estão organizados e as operações usadas por um programa para processar (ler ou escrever) esse arquivo.

A extensão só serve para o SO saber qual programa é o mais indicado para tal arquivo.

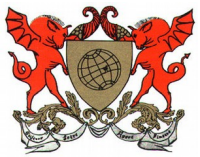


Formato de um Arquivo

Os arquivos podem ser classificados em dois tipos:

- Arquivos texto (ou formatados): podem ser editados no bloco de notas

- **Arquivos binários** (ou não-formatados): Não podem ser editados no bloco de notas

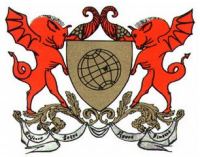


Formato de um Arquivo

Arquivos texto:

- Os dados são gravados exatamente como seriam impressos na tela;
- Os dados são gravados como vários caracteres de 8 bits utilizando a tabela ASCII.

Para isso, existe uma etapa de "conversão" dos dados;



Formato de um Arquivo

Arquivos texto:

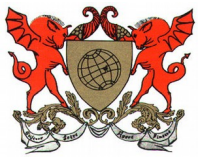
A conversão possui alguns problemas pois gera **arquivos maiores** e com isso a **leitura/escrita** ficam **mais lentas**.

Ex: Considere um número inteiro com 8 dígitos.

```
int n = 12345678; //32 bits na memória.
```

Num arquivo texto, cada dígito será convertido para seu caractere ASCII, ou seja, 8 bits por dígito.

```
12345678 //64 bits no arquivo.
```

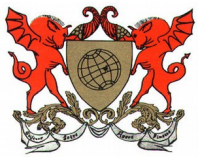


Formato de um Arquivo

Arquivos binários:

- Os dados são gravados exatamente como estão organizados na memória do computador;
- Não existe etapa de "conversão" dos dados.

Logo, os arquivos são em geral, **menores** e a **leitura/escrita são mais rápidas**.



Formato de um Arquivo

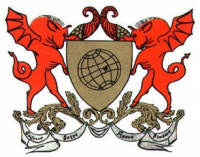
Arquivos binários:

Ex: Voltando ao inteiro de 8 dígitos.

```
int n = 12345678; //32 bits na memória.
```

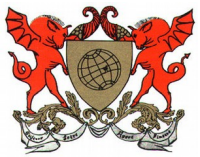
Num arquivo binário, o conteúdo da memória será copiado diretamente para o arquivo, sem conversão.

```
12345678 //32 bits no arquivo (codificado)
```



Arquivos

FORMATO	VANTAGENS	DESVANTAGENS
Texto	• Legibilidade • Pode ser editado com editor de texto • Arquivo pode ser facilmente transferido para outra plataforma	• Representação numérica pode não ser precisa • Pode ocupar muito mais espaço
Binário	• Não existe erro de conversão de valores numéricos • Leitura ou escrita de arquivos é mais rápida • Pode ocupar menos espaço	• Transferência para outra plataforma pode ser problemático



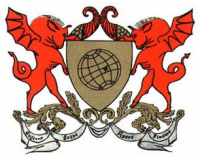
Arquivos (Declaração)

Biblioteca: #include<stdio.h>

A linguagem C usa um tipo especial de ponteiro para manipular arquivos. Neste caso é usado o tipo de dados **FILE**.

Forma geral:

```
FILE* nome_do_ponteiro;
```

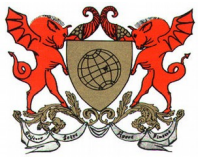


Arquivos (Declaração)

```
FILE* nome_do_ponteiro;
```

É esse ponteiro que controla o fluxo de leitura e escrita dentro de um arquivo;

Ou seja, ele serve para identificar um arquivo no disco, permitindo a gravação, alteração, exclusão de dados do arquivo.



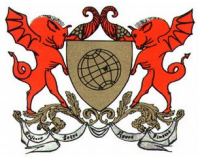
Manipulação de Arquivos

Basicamente são **três** as etapas para manipulação de arquivos:

(1) Abrir o arquivo;

(2) Ler e/ou gravar os dados desejados no arquivo;

(3) Fechar o arquivo.



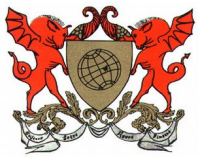
Manipulação de Arquivos

Abrindo Arquivo:

Para abrir um arquivo usamos a função **fopen()**:

```
FILE* fopen(char* nome, char* modo);
```

A função permite abrir um arquivo em um determinado modo de leitura ou escrita



Manipulação de Arquivos

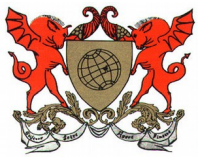
Abrindo Arquivo:

Para abrir um arquivo usamos a função fopen():

```
FILE* fopen(char* nome, char* modo);
```

fopen() recebe como parâmetro o **nome do arquivo** (string de C) e o **modo de abertura** (string de C).

Ela retorna um apontador (**FILE***) indicando o endereço do arquivo ou **NULL** se houver algum problema na abertura do arquivo.



Manipulação de Arquivos

Abrindo Arquivo:

```
#include <stdio.h>

int main(){
    FILE* f;
    f = fopen("arquivo.txt", "w");
    if(f == NULL){
        printf("Erro na abertura!\n");
        exit(1);//aborta o programa
    }
    return 0;
}
```

Manipulação de Arquivos

Abrindo Arquivo:

```
FILE* f;
f = fopen("arquivo.txt", "w");
```

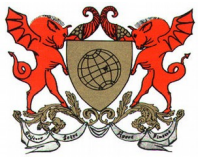
[onde guardar] *[nome]* *[tipo]*

Para gerar um código de programa que abre um arquivo, o compilador precisa conhecer 3 coisas:

1- **Onde guardar** informações sobre o arquivo

2- O **nome** do arquivo

3- O **tipo/modo** de abertura



Manipulação de Arquivos

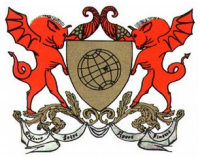
Abrindo Arquivo (Modos de Abertura):

Existem 3 tipos de abertura de arquivo:

- **“r”** para leitura (**read**)

- **“w”** para gravação (**write**)

- **“a”** para adicionar dados (**append**)

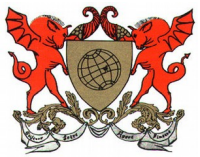


Manipulação de Arquivos

Abrindo Arquivo (Modos de Abertura):

Um modificador pode ser usado junto ao tipo:

- "b" para **modo binário**

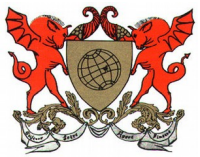


Manipulação de Arquivos

Abrindo Arquivo (Modos de Abertura):

- Arquivo Texto

Modo	Arquivo	Função
"r"	Texto	Leitura. Arquivo deve existir.
"w"	Texto	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a"	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").



Manipulação de Arquivos

Abrindo Arquivo (Modos de Abertura):

- Arquivo Binário

Modo	Arquivo	Função
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").

Manipulação de Arquivos

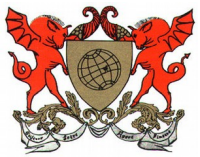
Abrindo Arquivo:

Cuidados ao abrir arquivos.

```
f = fopen("arquivo.txt", "w");
if(f == NULL){
   printf("Erro na abertura!\n");
   exit(1);//aborta o programa
}else
   printf("Arquivo aberto corretamente.\n");
```

Problemas:

- Gravação: não tem espaço em disco
- Leitura: o arquivo não existe



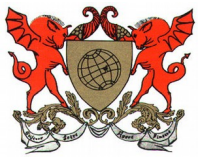
Manipulação de Arquivos

Abrindo Arquivo:

Cuidados ao abrir arquivos.

```
f = fopen("arquivo.txt", "w");
if(f == NULL){
   printf("Erro na abertura!\n");
   exit(1);//aborta o programa
}else
   printf("Arquivo aberto corretamente.\n");
```

Recomenda-se **SEMPRE** verificar se o arquivo foi aberto com sucesso, antes de dar sequência no programa.



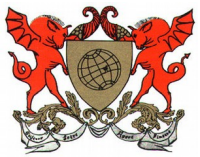
Manipulação de Arquivos

Fechando Arquivo:

Sempre que terminarmos de usar um arquivo, devemos fechá-lo. Para realizar essa tarefa, usa-se a função **fclose()**:

```
int fclose(FILE* f);
```

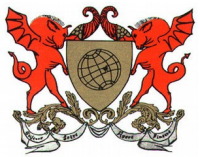
A função recebe como parâmetro um ponteiro f devolvido por fopen() e retorna ZERO no caso de sucesso no fechamento do arquivo.



Manipulação de Arquivos

Fechando Arquivo:

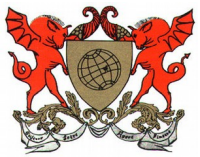
```
int main()
{
    FILE *f;
    f= fopen("arquivo.dat", "w");

    ...

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



Manipulação de Arquivos

Fechando Arquivo:

```
int main()
{
    FILE *f;
    int erro;
    f= fopen("arquivo.dat", "w");
    erro= fclose(f);
    if (erro == 0)
       printf("\nArquivo fechado com sucesso.\n");
    else
       printf("\nErro no fechamento.\n");
    return 0;
 }
```



Leitura e Gravação de dados em Arquivo

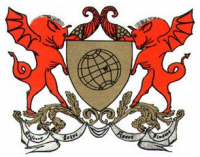
Formas de acessar arquivos:

1) Dados podem ser lidos e escritos **um caracter por vez**.

2) Dados podem ser lidos e escritos como **strings**.

3) Dados podem ser lidos e escritos de **modo formatado**.

4) Dados podem ser lidos e escritos em um formato chamado **registro ou bloco**. **(Formato Binário)**



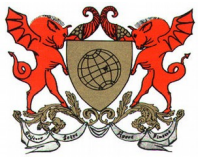
Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

1) Dados podem ser lidos e escritos **um caracter por vez**.

Existem duas funções:

- **fputc()**
- **fgetc()**



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

1) Dados podem ser lidos e escritos **um caracter por vez**.

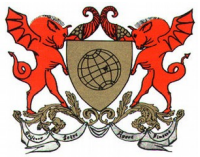
fputc():

```
int fputc(char c, FILE* f);
```

fputc() grava um caracter no arquivo.

onde, **c** é o caracter a ser gravado

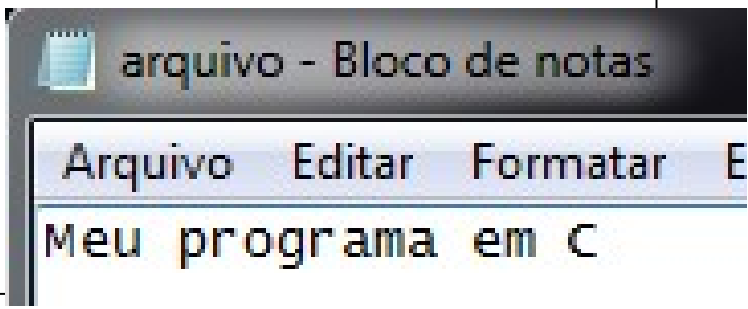
f é o ponteiro devolvido por fopen()



Leitura e Gravação de dados em Arquivo

fputc():

```
int main(){
     FILE* f;
     f = fopen("arquivo.txt", "w");
     if(f == NULL){
          printf("Erro na abertura.\n");
          exit(1);
     }
     char texto[20] = "Meu programa em C";
     int i;
     for(i=0; i<strlen(texto); i++)
        fputc(texto[i], f);
     fclose(f);
     return 0;
}
```



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

1) Dados podem ser lidos e escritos **um caracter por vez**.

fgetc():

```
int fgetc(FILE* f);
```

fgetc() lê um caracter do arquivo na posição em que o ponteiro de leitura estiver e move o ponteiro de leitura uma posição para frente.

onde, **f** é o ponteiro devolvido por fopen()

Leitura e Gravação de dados em Arquivo

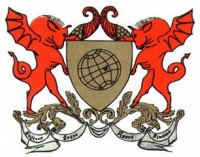
Formas de acessar arquivos:

1) Dados podem ser lidos e escritos **um caracter por vez**.

fgetc():

```
int fgetc(FILE* f);
```

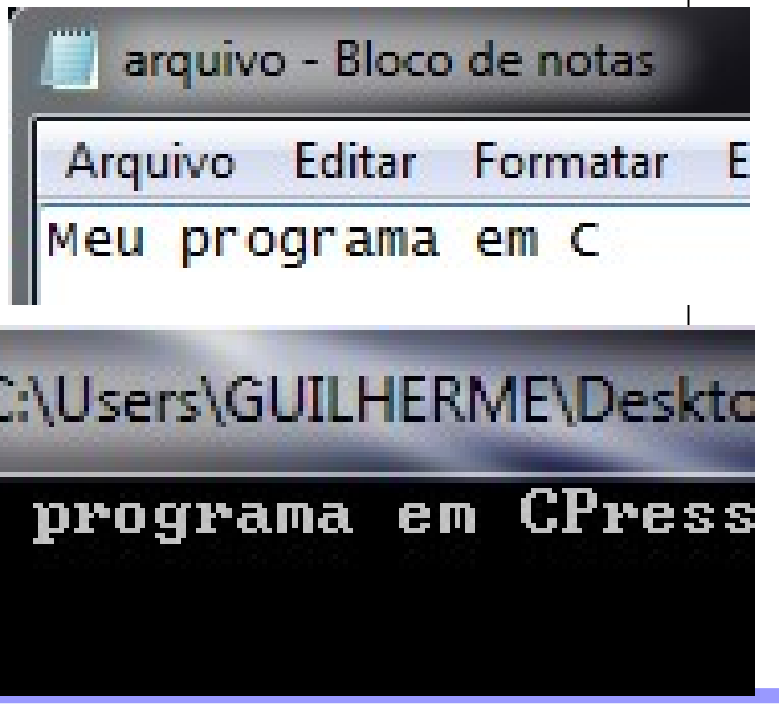
fgetc() retorna o valor inteiro da tabela ASCII do caracter lido.



Leitura e Gravação de dados em Arquivo

fgetc():

```
int main(){
     FILE* f;
     f = fopen("arquivo.txt", "r");
     if(f == NULL){
          printf("Erro na abertura.\n");
          exit(1);
     }
     char c = fgetc(f);
     while(c != EOF){
        printf("%c", c);
        c = fgetc(f);
     }
     fclose(f);
     return 0;
}
```



Leitura e Gravação de dados em Arquivo

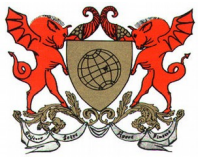
Constante EOF (End of file)

Significa “fim de arquivo”.

Essa constante é retornada por fgetc() quando tenta ler além do final de um arquivo.

Indica que o final do arquivo foi atingido.

Pode ser usada com arquivos texto, porém não pode ser usada com arquivos binários.



Faça um programa em C que imprima na tela o seu próprio código.

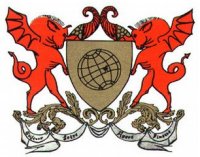
```c
#include <stdio.h>
#include <locale.h>

int main(){
    setlocale(LC_ALL,""); // caracteres com acentos no console
    FILE *arquivo;
    char ch;

    arquivo = fopen("autoprint.c", "r");
    ch = fgetc(arquivo);

    while (ch != EOF){
        printf("%c", ch);
        ch = fgetc(arquivo);
    }

    fclose(arquivo);

    return 0;
}
```

Leitura e Gravação de dados em Arquivo

Função rewind():

```
void rewind(FILE* f);
```

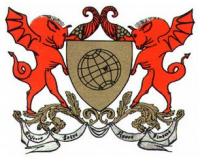
Essa função reinicia o arquivo, ou seja, movimenta o ponteiro do arquivo para seu início.



Leitura e Gravação de dados em Arquivo

Função rewind():

```
FILE *f;
f = fopen("arquivo.txt", "r"));
char ch;
ch = fgetc(f);
while (ch != EOF)
  ch = fgetc(f);
rewind(f);
```



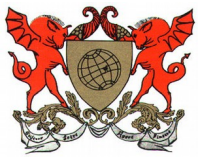
Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

2) Dados podem ser lidos e escritos **strings**.

Existem duas funções:

- **fputs()**
- **fgets()**



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

2) Dados podem ser lidos e escritos **strings**.

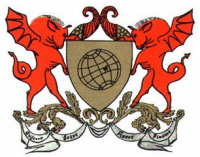
fputs():

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

fputs() grava uma string no arquivo.

onde, **s** é a string a ser gravada

f é o ponteiro devolvido por fopen()



Leitura e Gravação de dados em Arquivo

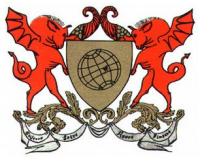
Formas de acessar arquivos:

2) Dados podem ser lidos e escritos **strings**.

fputs():

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

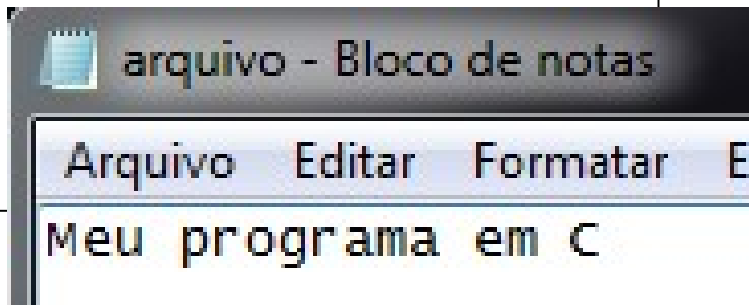
fputs() retorna a constante EOF em caso de erro e um valor diferente de ZERO em caso de sucesso.



Leitura e Gravação de dados em Arquivo

fputs():

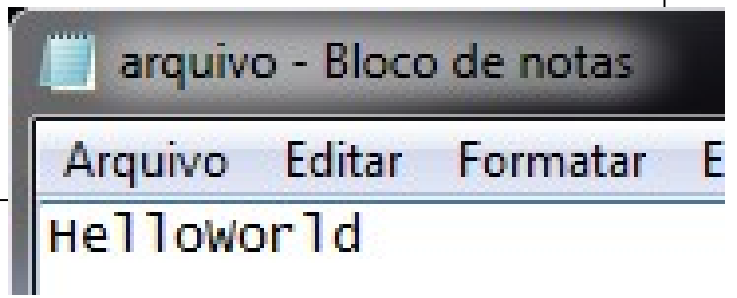
```
int main(){
     FILE* f;
     f = fopen("arquivo.txt", "w");
     if(f == NULL){
          printf("Erro na abertura.\n");
          exit(1);
     }
     char texto[20] = "Meu programa em C";
     int retorno = fputs(texto, f);
     if(retorno == EOF)
        printf("Erro na Gravacao.\n");
     fclose(f);
     return 0;
}
```



Leitura e Gravação de dados em Arquivo

fputs():

```
int main(){
     FILE* f;
     f = fopen("arquivo.txt", "w");
     if(f == NULL){
          printf("Erro na abertura.\n");
          exit(1);
     }
     fputs("Hello", f);
     fputs("World", f);
     fclose(f);
     return 0;
}
```



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

2) Dados podem ser lidos e escritos **strings**.

fgets():

```
char* fgets(char* s, int tam, FILE* f);
```

fgets() lê uma linha por vez do arquivo.

onde, **s** é a string que recebe a string lida do arquivo.
tam é o tamanho máximo da string s ou,
quantos caracteres no máximo serão lidos.
f é o ponteiro devolvido por fopen()

Leitura e Gravação de dados em Arquivo

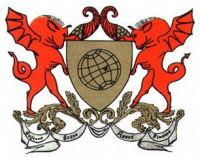
Formas de acessar arquivos:

2) Dados podem ser lidos e escritos **strings**.

fgets():

```
char* fgets(char* s, int tam, FILE* f);
```

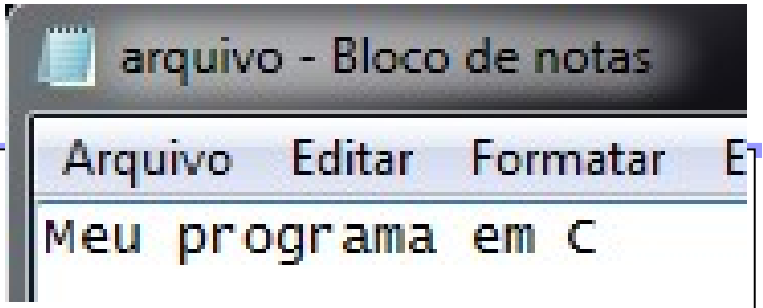
fgets() retorna NULL em caso de erro. Em caso de sucesso, ela retorna um ponteiro para o primeiro caractere de s.

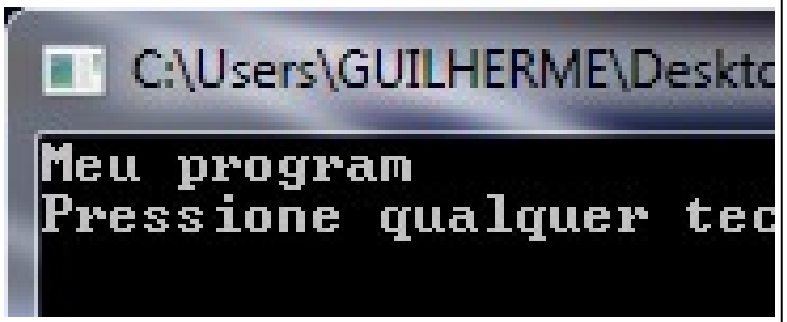


Leitura e Gravação de dados em Arquivo

fgets():

```
int main(){
     FILE* f;
     f = fopen("arquivo.txt", "r");
     if(f == NULL){
          printf("Erro na abertura.\n"); exit(1);
     }
     char texto[20];
     char* result = fgets(texto, 12, f);//11 caracteres
     if(result == NULL)
        printf("Erro na Leitura.\n");
     else
        printf("%s", texto);
     fclose(f);
     printf("\n");
     return 0;
}
```





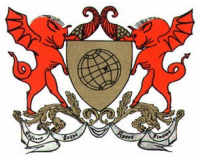
Leitura e Gravação de dados em Arquivo

fgets():

Lê caracteres até atingir um **caractere '\n'**, ou o **final do arquivo** ou o **número máximo de caracteres especificado**.

Escreve um caractere nulo '\0' após o último caractere armazenado no vetor.

Quando o final do arquivo é atingido ao tentar ler novamente ela retorna **NULL**;



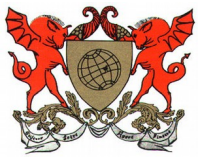
Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

3) Dados podem ser lidos e escritos **de modo formatado**.

Existem duas funções:

- **fprintf()**
- **fscanf()**



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

3) Dados podem ser lidos e escritos **de modo formatado**.

printf():

```
printf(char* s, variaveis);
```

fprintf():

```
fprintf(FILE* f,char* s, variaveis);
```

onde, **s** é a string que será gravada no arquivo

f é o ponteiro devolvido por fopen()

Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

3) Dados podem ser lidos e escritos **de modo formatado**.

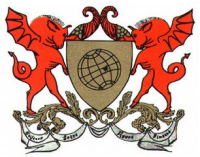
printf():

```
printf(char* s, variaveis);
```

fprintf():

```
fprintf(FILE* f,char* s, variaveis);
```

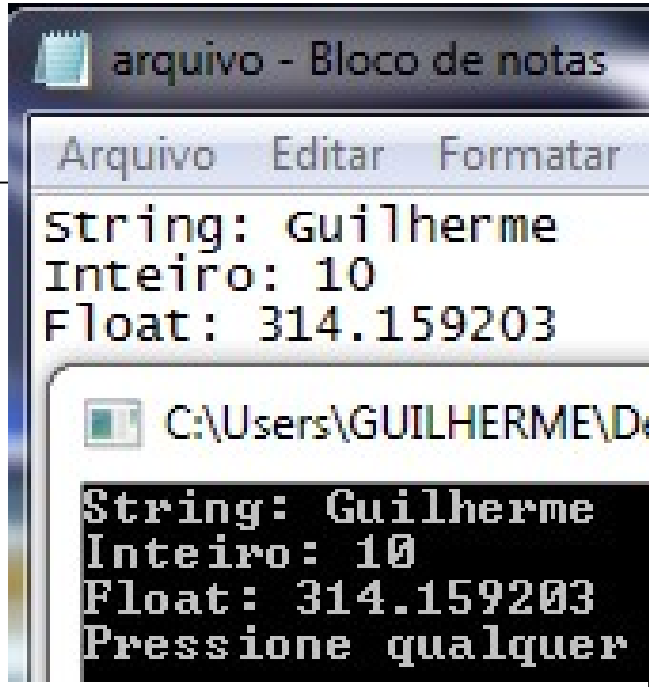
fprintf() grava os dados no arquivo da mesma forma que seriam mostrados na tela com a função printf().



Leitura e Gravação de dados em Arquivo

fprintf():

```
int main(){
   FILE* f;
   f = fopen("arquivo.txt", "w");
   if(f == NULL){
        printf("Erro na abertura.\n");exit(1);
   }
   char nome[30] = "Guilherme";
   int r = 10;
   float pi = 3.141592;
   printf("String: %s\nInteiro: %d\nFloat: %f\n",  nome, r, pi*r*r);
   fprintf(f,"String: %s\nInteiro: %d\nFloat:%f\n", nome, r, pi*r*r);
   fclose(f);
   system("pause");
   return 0;
}
```



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

3) Dados podem ser lidos e escritos **de modo formatado**.

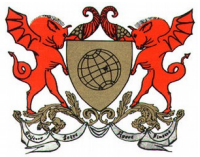
scanf():

```
scanf(char* s, variaveis);
```

fscanf():

```
fscanf(FILE* f,char* s, variaveis);
```

onde, **s** é a string dos **tipos de entrada (%d, %s, %f, ...)**
f é o ponteiro devolvido por fopen()



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

3) Dados podem ser lidos e escritos **de modo formatado**.

scanf():

```
scanf(char* s, variaveis);
```

fscanf():

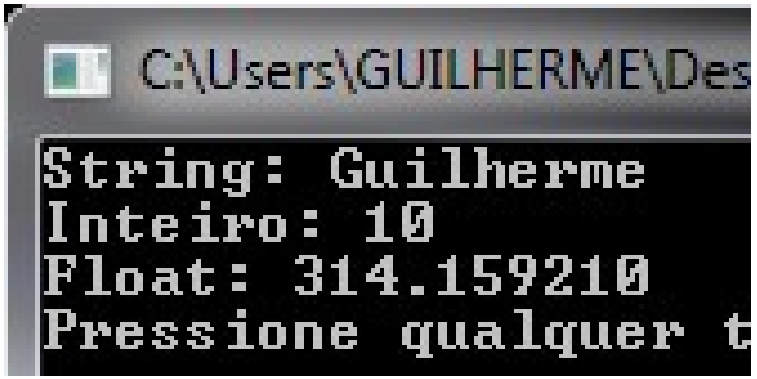
```
fscanf(FILE* f,char* s, variaveis);
```

fscanf() lê os dados do arquivo da mesma forma que seriam lidos do teclado com a função scanf().

Leitura e Gravação de dados em Arquivo

fscanf():

```
int main(){
   FILE* f;
   f = fopen("arquivo.txt", "r");
   if(f == NULL){
        printf("Erro na abertura.\n");exit(1);
   }
   char nome[30], texto[20];
   int r; float x;
   fscanf(f,"%s %s",texto, nome);
   printf("%s %s\n",texto, nome);
   fscanf(f,"%s %d",texto, &r);
   printf("%s %d\n",texto, r);
   fscanf(f,"%s %f",texto, &x);
   printf("%s %f\n",texto, x);
   fclose(f);
   system("pause");
   return 0;
}
```



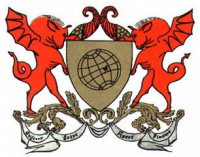
Exercício

1) Crie um programa que grava em um arquivo de dados, os nomes e idades de várias pessoas.

Esses nomes e idades a serem gravados deverão ser digitados pelo usuário e no arquivo final formatado da seguinte maneira:

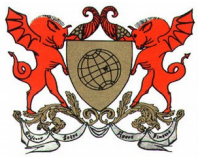
A primeira linha contém um inteiro n informando quantas pessoas. As n linhas seguintes contém as informações das pessoas.

```
4
Carlos 14
Antonio 16
Jose 30
Moises 25
```



Exercício

2) Crie um outro programa que lê o arquivo do exercício anterior e mostre na tela o nome da pessoa mais velha.



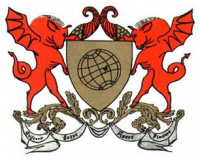
Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

4) Dados podem ser lidos e escritos em um formato chamado **registro ou bloco**. **(Formato Binário)**

Existem duas funções para escrita e leitura de blocos de bytes:

- **fwrite()**
- **fread()**



Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

4) Dados podem ser lidos e escritos em um formato chamado **registro ou bloco**. **(Formato Binário)**

fwrite():

```
int fwrite(void* buffer,int bytes,
int total, FILE* f);
```

fwrite() grava um bloco de bytes em um arquivo.

onde, **buffer** é um ponteiro para a localização na memória dos dados

bytes é o tamanho, em bytes, de cada unidade a ser gravada

total é o total de unidades de dados que devem ser gravadas

f é o ponteiro devolvido por fopen()

Leitura e Gravação de dados em Arquivo

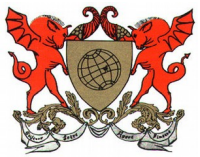
Formas de acessar arquivos:

4) Dados podem ser lidos e escritos em um formato chamado **registro ou bloco**. **(Formato Binário)**

fwrite():

```
int fwrite(void* buffer,int bytes,
int total, FILE* f);
```

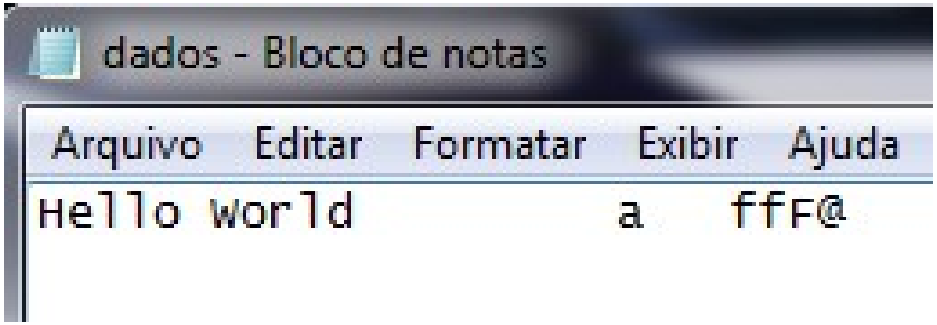
fwrite() retorna o total de unidades de dados gravadas com sucesso.



Leitura e Gravação de dados em Arquivo

fwrite():

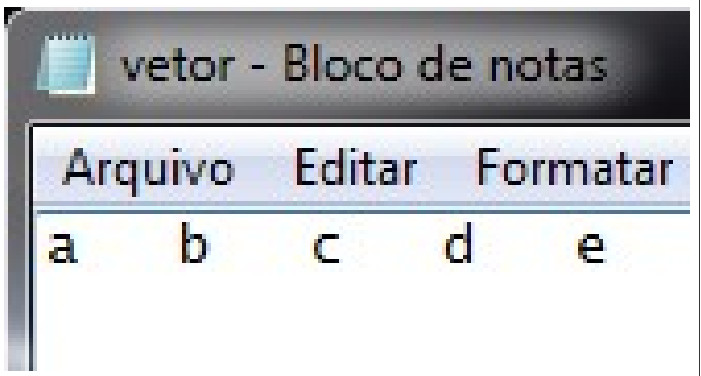
```
int main(){
   FILE* f;
   f = fopen("vetor.txt", "wb");
   if(f == NULL){
        printf("Erro na abertura.\n");exit(1);
   }
   char str[20] = "Hello World";
   int a = 97;
   float x = 3.1;
   fwrite(str, sizeof(char), 20, f);
   fwrite(&a, sizeof(int), 1, f);
   fwrite(&x, sizeof(float), 1, f);
   fclose(f);
   return 0;
}
```



Leitura e Gravação de dados em Arquivo

fwrite():

```
int main(){
   FILE* f;
   f = fopen("vetor.txt", "wb");
   if(f == NULL){
        printf("Erro na abertura.\n");exit(1);
   }
   int v[5] = {97, 98, 99, 100, 101}, total;
   total = fwrite(v, sizeof(int), 5, f);
   if(total != 5) {
        printf("Erro na escrita.\n");exit(1);
   }
   fclose(f);
   system("pause");
   return 0;
}
```



ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	*[DEL]*

Leitura e Gravação de dados em Arquivo

Formas de acessar arquivos:

4) Dados podem ser lidos e escritos em um formato chamado **registro ou bloco**. **(Formato Binário)**

fread():

```
int fread(void* buffer,int bytes,
int total, FILE* f);
```

fread() lê um bloco de bytes de um arquivo.

onde, **buffer** é um ponteiro para a localização na memória dos dados

bytes é o tamanho, em bytes, de cada unidade a ser lida

total é o total de unidades de dados que devem ser lidas

f é o ponteiro devolvido por fopen()

Leitura e Gravação de dados em Arquivo

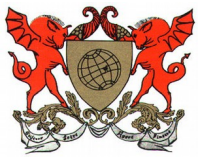
Formas de acessar arquivos:

4) Dados podem ser lidos e escritos em um formato chamado **registro ou bloco**. **(Formato Binário)**

fread():

```
int fread(void* buffer,int bytes,
int total, FILE* f);
```

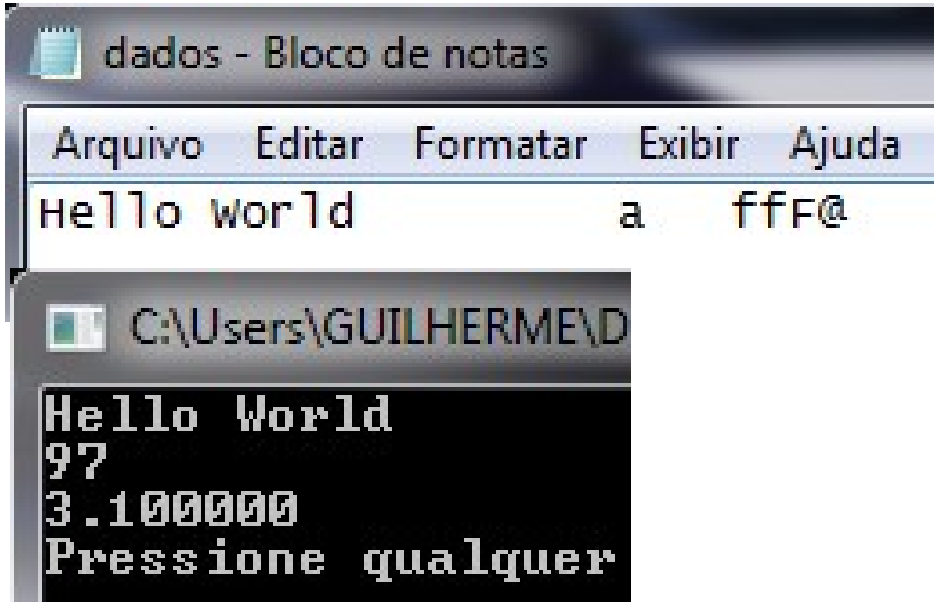
fread() retorna o total de unidades de dados lidos com sucesso. Quando é 0, pode indicar o fim de arquivo ou que ocorre algum erro antes da leitura de algum elemento.



Leitura e Gravação de dados em Arquivo

fread():

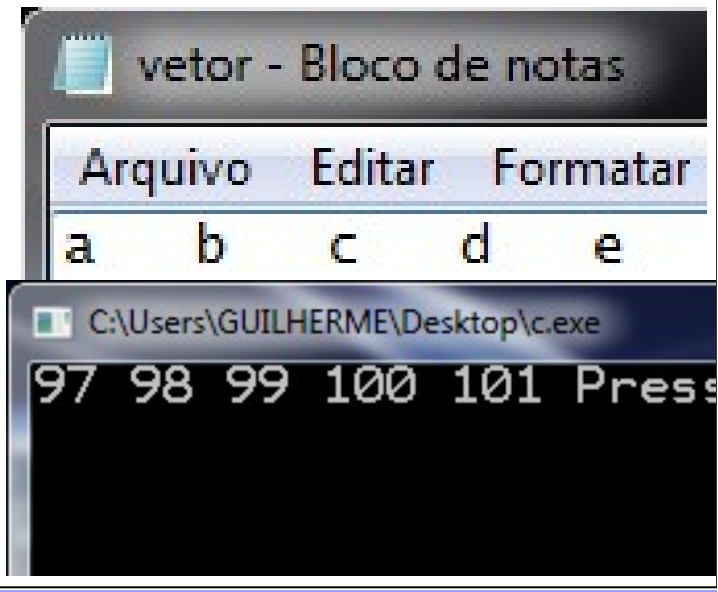
```
int main(){
  FILE* f;
  f = fopen("dados.txt", "wb");
  if(f == NULL){
       printf("Erro na abertura.\n");exit(1);
  }
  char str[20];
  int a;
  float x;
  fread(str, sizeof(char), 20, f);
  fread(&a, sizeof(int), 1, f);
  fread(&x, sizeof(float), 1, f);
  printf("%s\n%d\n%f\n",str, a, x);
  fclose(f);
  return 0;
}
```



Leitura e Gravação de dados em Arquivo

fread():

```
int main(){
  FILE* f;
  f = fopen("vetor.txt", "rb");
  if(f == NULL){
       printf("Erro na abertura.\n");exit(1);
  }
  int v[5], total, i;
  total = fread(v, sizeof(int), 5, f);
  if(total != 5) {
       printf("Erro na leitura.\n");exit(1);
  }
  for(i=0; i<5; i++) printf("%d ",v[i]);
  fclose(f);
  system("pause");
  return 0;
}
```

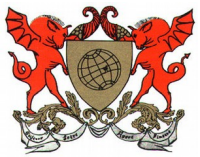


Exercícios

3) a. Elabore um programa em C que grave em um arquivo texto um int, um float, um char, uma palavra, um vetor de inteiros, uma frase, um vetor de struct (Pessoa).

b. Elabore outro programa que leia todos esses dados do arquivo.

4) Faça o mesmo do exercício anterior usando arquivos binários, grave e leia.



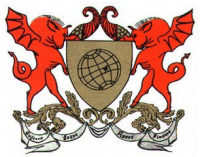
Exercícios

5) Elabore um programa em C que leia os seguintes campos para o cadastro de um fornecedor:

- Nome
- Numero de produtos
- Endereço
- Telefone

Leia alguns fornecedores e grave os dados no arquivo "forn.txt".

Elabore outro programa em C que apresente na tela todos os fornecedores armazenados no arquivo "forn.txt".



Exercícios

6) Usando a estrutura do exercício anterior, aloque um vetor de fornecedores, preencha-o através de uma **função não-recursiva** e grave os dados em um arquivo **binário**.

Usando o arquivo, crie outro programa que lê estes dados armazenados e imprima-os na tela usando uma **função recursiva**.

Além disso, imprima no final o nome do fornecedor com a maior quantidade de produtos.