

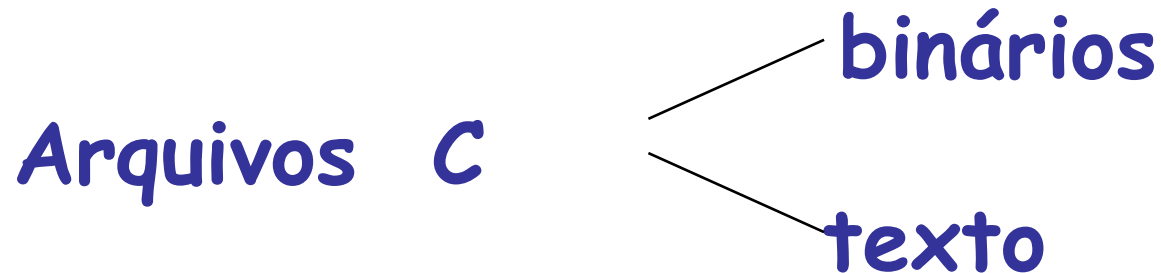
INF101202

Algoritmos e Programação

Modalidade Ead – Turma H

Material de apoio: arquivos texto
Processamento de arquivo texto:
linha a linha e caractere a caractere

Arquivos:



Esta apresentação versa sobre arquivos texto:
são aqueles em que as informações estão
codificadas como caracteres ASCII.

Arquivo: relembrando.....

A linguagem C não impõe estrutura alguma aos arquivos e, do ponto de vista físico, um arquivo é, simplesmente, uma sequência (ou *fluxo*) de *bytes*.

Mas, do ponto de vista lógico, nós, os projetistas, imaginamos os dados armazenados em variáveis ou organizados em estruturas (*struct*), vetores, matrizes, etc.

Arquivo: operações para manipulação

Modos de abertura para arquivo Texto:

"**r**" - read, abertura do arquivo para leitura; se o arquivo não existir ocorrerá um **erro**!

"**w**" - write, abertura do arquivo para escrita; se o arquivo já existir, a gravação **ocorrerá sobre** os dados existentes, ou seja, é criado um novo, com perda do anterior;

"**a**" -append, abertura do arquivo para acrescentar novos dados. Se o arquivo não existir, é criado um novo e funciona tal qual o modo "w"; mas se já existir, os novos dados são acrescentados no final do arquivo, a partir dos já existentes e de modo sequencial.

```
arq = fopen("dadosmeus.txt","a");
```

```
pf = fopen("dadosteus.txt","w");
```

Arquivo: operações para manipulação

Modos de abertura para arquivo Texto:

"**r+**" - abre um arquivo texto para leitura/escrita; se o arquivo não existir, ele é criado;

"**w+**" - cria um arquivo texto para escrita/leitura; se o arquivo já existir, ele é recriado, ou seja, seu conteúdo anterior é apagado;

"**a+**" - acrescenta dados no final para a escrita; se o arquivo não existir, ele é criado.

```
arq = fopen("dadosmeus.txt","a+");  
pf = fopen("dadosteus.txt","w+");
```

Arquivo Texto

- elemento/posição corrente:
 - um caractere em uma linha
- marcador de final de linha:
 - nos programas em C: '\n'
 - em Unix, Linux, AIX, Xenix, Mac OS X, BeOS, ...: LF (line feed)
 - em Commodore, Apple II e até Mac OS 9: CR (carriage return)
 - em DOS, OS/2, Microsoft Windows: CR+LF

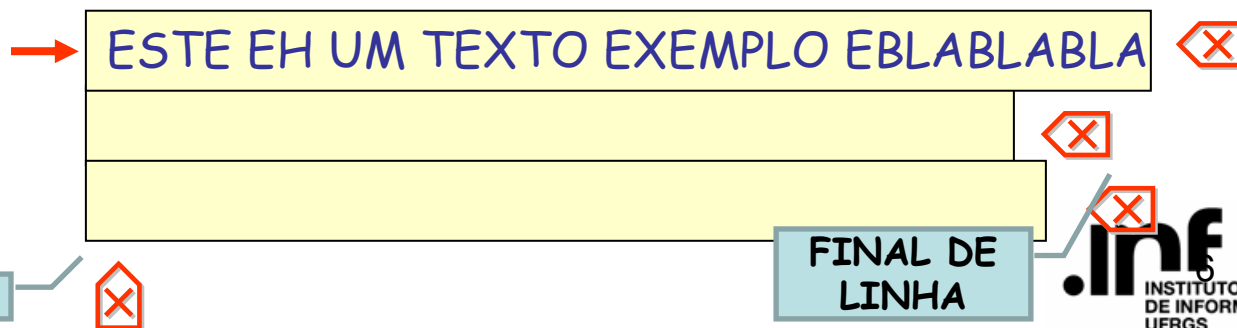
Obs.: o sistema operacional converte o \n para seu padrão de fim de linha!

- final de arquivo:

- EOF (ou -1)



arq.txt



Arquivo Binário vs Arquivo Texto

ARQUIVO BINÁRIO

- dados: podem ser estruturados e ou simples (estruturas, inteiros, etc.)
- organizado em blocos de bytes;
- cada elemento pode corresponder a um ou mais bytes;
- valores numéricos são representados em forma binária.

ARQUIVO TEXTO

- dados: somente caracteres (cada caractere = um byte);
- organizado em sequências de caracteres, formando linhas de tamanho variável.
- Todos os tipos de valores são armazenados como sequências de caracteres. No caso de valores reais, o ponto decimal e o sinal, se existir, também são armazenados.

Exemplificando as diferenças entre arquivos binários e arquivos texto:

A seguir, a saída de dois programas que geram arquivos com *floats*:

- o primeiro gerou um arquivo binário de floats;
- o segundo gerou um arquivo texto, com cada linha contendo um valor float. Nesse caso, na escrita dos valores foi usado o formato `%6.2f`.

Observar os valores retornados por `ftell` antes e depois da escrita em cada um dos arquivos.

Eles deixam claro que enquanto no arquivo binário sempre são escritos 4 bytes por valor *float*, no arquivo texto o número de *bytes* é dependente dos valores *float* informados em relação ao formato utilizado.

Os valores em que o número de caracteres a serem apresentados ultrapassa 6 são gravados utilizando mais do que seis *bytes*.

Geração de um arquivo binário de floats

```
C:\ C:\backupcida\LinguagemCPagina20082\Aula25\criarqbinariofloat.e
Nome do arquivo (com no maximo 29 caracteres): c:\float
ftell antes da escrita = 0
Valor: 12345.5678
ftell depois da escrita = 4
1-InserirNovo, 2-Encerrar
1
ftell antes da escrita = 4
Valor: 12345678.234
ftell depois da escrita = 8
1-InserirNovo, 2-Encerrar
1
ftell antes da escrita = 8
Valor: 1
ftell depois da escrita = 12
1-InserirNovo, 2-Encerrar
1
ftell antes da escrita = 12
Valor: 2.3
ftell depois da escrita = 16
1-InserirNovo, 2-Encerrar
2
Pressione qualquer tecla para continuar. . .
```

Geração de um arquivo texto com valores float, usando formato %6.2f

```
C:\ C:\backupcida\LinguagemCPagina20082\Aula25\criaarqtextofloatcomfo
Nome do arquivo (com no maximo 29 caracteres): c:\float2
ftell antes escrita = 0
Valor: 123456.499999
ftell depois escrita = 9
ftell depois do \n = 11
1-InserirNovo, 2-Encerrar
1
ftell antes escrita = 11
Valor: 123456.444444
ftell depois escrita = 20
ftell depois do \n = 22
1-InserirNovo, 2-Encerrar
1
ftell antes escrita = 22
Valor: -123456.49999
ftell depois escrita = 32
ftell depois do \n = 34
1-InserirNovo, 2-Encerrar
2
Pressione qualquer tecla para continuar. . .
```

Valores fornecidos excedem as seis posições do formato, então são armazenados o ponto, duas casas decimais (com arredondamento) e todas as casas inteiras = 9 posições

Neste caso ainda há o sinal, logo = 10 posições

Funções para processar arquivos texto - linha a linha:

Leitura de uma linha:

`fgets (&strbuffer,tamanho,ponteiro para arquivo)`

Lê de um arquivo uma *string* de um tamanho máximo informado.

Para leitura de *strings* a partir do dispositivo padrão de entrada, o formato geral de *fgets* é:

`fgets(string (=vetor de caracteres), tamanho máximo da string, stdin)`
para outros arquivos, em vez de *stdin*, aparece o ponteiro do arquivo:

`fgets(string (=vetor de caracteres), tamanho máximo da string, ponteiro do arquivo);`

A *string* é lida até o tamanho máximo indicado - 1 (para inserção do '\0').

O caractere de nova linha \n, se dentro do tamanho máximo previsto, também integrará a *string*.

Em algumas situações pode ser interessante eliminar o \n, se ele estiver presente no final da string, para que não interfira por exemplo em comparações entre strings.

Funções para processar arquivos texto linha a linha:

Escrita de uma linha:

`fputs (&strbuffer, ponteiro para arquivo)`

Escreve uma string/linha num arquivo.
Não grava '\0' no arquivo, apenas '\n', se existir.

Ex.: `fputs (linha, arq2);`

Geração de um arquivo texto linha a linha

Gravar um arquivo texto com linhas fornecidas pelo usuário via teclado. Uma vez criado o arquivo, apresentar seu conteúdo.

Geração do arquivo e listagem do arquivo deverão ser realizadas linha a linha.

Geração de um arquivo texto linha a linha

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLIN 80
int grava_arq(FILE *, int);
int le_arq(FILE *, int);
int main( )
{
    FILE *arq;
    int linhas_grav = 0, linhas_lid = 0;
    system("color f1");
    if ((arq = fopen("c:\\texto01", "w")) == NULL)
    {
        printf("Erro abertura arquivo para gravar\n");
        system("pause");
    }
    else
    {
        ...
    }
}
```

cont.

Geração de um arquivo texto linha a linha

```
...
linhas_grav = grava_arq(arq, MAXLIN);
printf("\nLinhas gravadas = %d\n", linhas_grav);
fclose(arq);
if ((arq = fopen("c:\\texto01", "r")) == NULL)
{
    printf("Erro abertura arquivo para ler\n");
    system("pause");
}
else
{
    linhas_lid = le_arq(arq, MAXLIN);
    printf("\nLinhas lidas = %d\n", linhas_lid);
    fclose(arq);
    system("pause");
    return 0;
}
}
```

cont.

Geração de um arquivo texto linha a linha

```
int grava_arq(FILE *arq1, int maximo)
{
    char linha[maximo];
    int gravs = 0;
    printf("\nDigite quantas linhas de %d caracteres desejar\n", maximo);
    printf ("Para parar FIM no inicio de uma linha\n");
    fgets(linha, sizeof(linha), stdin);
    if (linha[strlen(linha) - 1] == '\n')
        linha[strlen(linha) - 1] = '\0';

    while ((strcmp(linha, "FIM")))
    {
        fputs (linha, arq1);
        putc('\n', arq1);
        gravs++;
        fgets(linha, sizeof(linha), stdin);
        if (linha[strlen(linha) - 1] == '\n')
            linha[strlen(linha) - 1] = '\0';
    }
    return gravs;
}
```

cont.

Geração de um arquivo texto linha a linha

```
int le_arq(FILE * arq2, int maximo)
{
    char linha[maximo];
    int lidas = 0, i;
    fgets(linha, sizeof(linha), arq2);
    for (i = 0; i < strlen(linha) ; i++)
        printf("\n%d\n", linha[i]);
    while (!(feof(arq2)))
    {
        printf("%s", linha);
        lidas++;
        fgets (linha, sizeof (linha) , arq2);
    }
    return lidas;
}
```

Geração de um arquivo texto linha a linha

```
F:\ARQUIVOS20091PARTE2\GravELeitArqLinhaALinha.exe

Digite quantas linhas de 80 caracteres desejar
Para parar FIM no inicio de uma linha
ouviram do
ipiranga
as margens
FIM

Linhas gravadas = 3

111
117
118
105
114
97
109
32
100
111
10
ouviram do
ipiranga
as margens

Linhas lidas = 3
Pressione qualquer tecla para continuar. . .
```

Os números ao lado, após a linha:

Linhas gravadas = 3

são os valores ASCII correspondentes aos caracteres da primeira linha digitada, que foram gravados no arquivo. 32 é o espaço em branco e 10 é a quebra de linha (\n). A quebra de linha deve ser inserida explicitamente no arquivo se múltiplas linhas nele forem desejadas.

```
texto01 - WordPad
Arquivo  Editar  Exibir  Inserir  Formatar  Ajuda

|ouviram do
|ipiranga
|as margens
```

Funções para processar arquivos texto caractere a caractere:

Escrita de um caractere

`putc (caractere, ponteiro para arquivo)`

Escreve um caractere no arquivo

Ex.: `putc(novo,arq);`

Leitura de um caractere

`getc (ponteiro para arquivo)`

Retorna um caracter lido do arquivo

Ex.: `while (!feof(arq))
printf ("%c", getc(arq));`

Geração de uma cópia de um arquivo texto

Ler um arquivo texto e gerar uma sua cópia.
Apresentar os arquivos original e cópia.

Obs.: no exemplo a seguir, o arquivo cópia é criado linha a linha, e a apresentação na tela dos conteúdos do arquivo original e da cópia são feitos caractere a caractere.

Geração de uma cópia de um arquivo texto

```
#include <stdio.h>
#include <stdlib.h>
#define MAXLINHA 255
void imprimearq (FILE *);
int main( )
{
    FILE *arq1;
    FILE *arq2;
    int cont = 0;
    char linha[MAXLINHA];
    system("color 70");
    if ((arq1 = fopen("texto02.txt", "r")) == NULL)
    {
        printf("Erro ao abrir arquivo entrada - 1 \n");
        system("pause");
    }
    else
        if ((arq2 = fopen("textosai1", "w")) == NULL)
        {
            printf("Erro ao abrir arquivo saida - 1 \n");
            system("pause");
        }
    else
    {
        (...)
    }
}
```

cont.

Geração de uma cópia de um arquivo texto

```
(...)  
while (!feof(arq1))  
    if (fgets(linha, sizeof(linha), arq1))  
    {  
        cont++ ;  
        fputs (linha, arq2);  
        fflush(arq2);  
    }  
printf("\nLinhas processadas: %d\n", cont);  
fclose (arq1);  
fclose(arq2);  
(...)
```

cont.

Geração de uma cópia de um arquivo texto

```
(...)  
if ((arq1 = fopen("texto02.txt", "r")) == NULL)  
{  
    printf("Erro ao abrir arquivo de entrada - 2 \n");  
    system("pause");  
}  
else  
{  
    printf("\n***ARQUIVO DE ENTRADA***\n");  
    imprimearq(arq1);  
    fclose(arq1);  
}  
if ((arq2 = fopen("textosai1", "r")) == NULL)  
{  
    printf("Erro ao abrir arquivo de saida - 2 \n");  
    system("pause");  
}  
else  
{  
    printf("\n\n***ARQUIVO DE SAIDA***\n");  
    imprimearq(arq2);  
    fclose(arq2);  
}  
}  
printf("\n\n");  
system("pause");  
return 0;
```

Slide 23

cont.

Geração de uma cópia de um arquivo texto

```
void imprimearq(FILE *arq)
{
    while (!feof(arq))
        printf ("%c", getc(arq));
}
```


Processamento de caracteres determinados em um arquivo texto (com uso de acesso randômico)

Ler um arquivo texto e trocar todas as ocorrências de um caractere no arquivo por outro caractere, informado pelo usuário em tempo de execução.

Processamento de caracteres determinados em um arquivo texto (com uso de acesso randômico)

Exemplo 2

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
    FILE *arq;
    char antigo, novo, temp, caract;
    int alt = 0;
    system("color f1");
    if ((arq = fopen("c:\\texto01", "r+")) == NULL)
    {
        printf("Erro ao abrir \n");
        system("pause");
    }
    else
    {
        printf("\nCaractere a procurar: \n");
        scanf(" %c", &antigo);
        printf("\nSubstituir por: \n");
        scanf(" %c", &novo);
        ...
    }
}
```

cont.

Processamento de caracteres determinados em um arquivo texto (com uso de acesso randômico)

```
...  
while (!feof(arq))  
{  
    temp = getc(arq);  
    if (temp == antigo)  
    {  
        alt++;  
        /*volta a posicao do caractere procurado*/  
        fseek(arq, -1*sizeof(char), SEEK_CUR);  
        putc(novo, arq); /*substitui*/  
        fflush(arq); /*escreve mudancas*/  
    }  
}  
fclose(arq);  
printf("\nNumero de caracteres alterados: %d\n", alt);
```

Troca de um caractere por outro

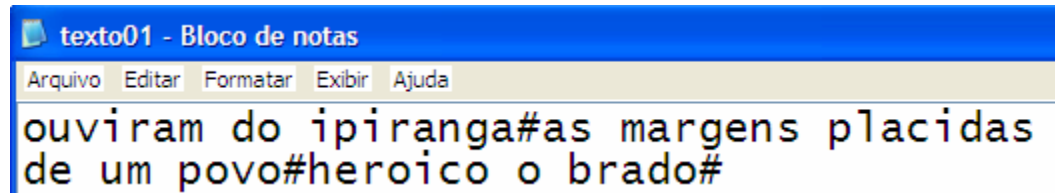
cont.

Processamento de caracteres determinados em um arquivo texto (com uso de acesso randômico)

```
...  
if ((arq = fopen("c:\\texto01.txt", "r")) == NULL)  
{  
    printf("Erro ao abrir para listar \n");  
    system("pause");  
}  
else  
{  
    printf("\n\n");  
    while (!feof(arq))  
        printf ("%c", getc(arq));  
    fclose(arq);  
    system("pause");  
    return 0;  
}  
}
```

Leitura de arquivo
texto caractere a
caractere

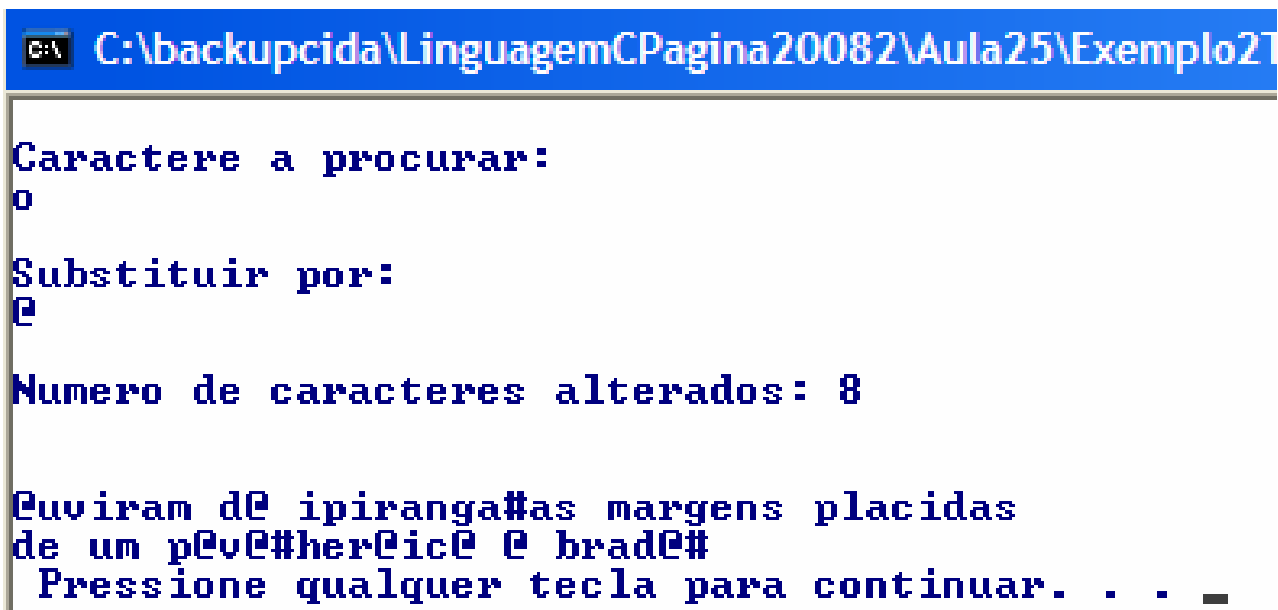
Processamento de caracteres determinados em um arquivo texto (com uso de acesso randômico)



texto01 - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

ouviram do ipiranga#as margens placidas
de um povo#heroico o brado#



C:\backupcida\LinguagemCPagina20082\Aula25\Exemplo2T

Caractere a procurar:
o

Substituir por:
@

Numero de caracteres alterados: 8

@uviram d@ ipiranga#as margens placidas
de um p@v@#her@ic@ @ brad@#
Pressione qualquer tecla para continuar. . . _

Arquivos: comparação entre modo texto e binário

Texto:

- **facilidade de criação:** para criar um arquivo texto basta usar um editor de textos, como o próprio editor do ambiente DEv-C++, ou o Bloco de Notas;
- **versatilidade** para se **verificar** o conteúdo, pois as informações estão em caracteres, o que permite verificação visual direta dos dados contidos nos arquivos;
- **processamento é lento**, pois os dados estão em ASCII e precisam ser convertidas para a codificação binária para serem armazenados na memória principal;
- **consome muita memória**, inviabilizando seu uso para grandes volumes de dados. Ex.: para armazenar-se o valor 10.00 em um arquivo texto precisa-se de 5 bytes, enquanto a representação binária de um float nas versões de C com as quais estamos trabalhando só exige 4 bytes.

Arquivos: comparação entre modo texto e binário

Binário:

- processamento é mais rápido, pois os dados já estão armazenados em codificação binária;
- consome **menos memória**, o que os tornam mais adequados para grandes volumes de dados;
- seu conteúdo não é identificável como no caso dos arquivos texto, sendo então mais adequado para armazenamento de informações sigilosas;
- nos arquivos binários a informação não sofre qualquer conversão/tradução.