Arquivos

- objetos *Persistentes x Temporários*
- Alto Nível (bufferizada) x Baixo Nível
- alto nível
  - um caracter por vez: putchar, getchar
  - strings: gets, puts
  - dados formatados: printf, scanf
  - blocos ou registros
- baixo nível
  - buffers mantidos pelo programador
- funções disponíveis
  - getc, putc
  - fgets, fputs
  - fscanf, fprintf
  - fread, fwrite

Texto x Binário

- texto: seqüência de caracteres agrupadas em linhas
  - C ~ separadas por um caracter **LF** (ASCII 10)
  - Dos ~ separadas por **CR** (13) + **LF** (10)
- compilador C
  - converte o par **CR/LF** para um único caracter **'\n'** na leitura
  - converte o caracter de nova linha **'\n'** no par **CR/LF** na gravação
- no modo texto
  - S.O. envia uma indicação de fim de arquivo durante na leitura do carater (1A)
  - EOF indica fim de arquivo
- **Cuidado**
  - EOF não é um carater mas um valor numérico
  - definido no arquivo stdio.h com valor -1
  - pode ter diferentes valores dependendo do S.O.

A Estrutura FILE

- analisando o arquivo stdio.h
  - ```
    #define  OPEN_MAX   20
    typedef  struct  {
      int            level;
      unsigned  flags;      /*  File status flags  */
      char           fd;        /* File descriptor      */
      unsigned char   hold;
      int            bsize;       /* Buffer size      */
      unsigned char   _FAR *buffer;
      unsigned char   _FAR *curp;    /* Current active pointer */
      unsigned       istemp;          /* Temporary file indicator   */
      short          token;             /* Used for validity checking */
    }  FILE;

    extern  FILE _streams [ OPEN_MAX ];
    ```

- funções
  - fopen( ): preenche inicialmente a estrutura FILE retornando um apontador para a mesma
    ```

Exemplo

```
#include  < stdio.h >
main( ) {
    FILE  *ptr;    int  ch;

    ptr  =  fopen( “arqtext.txt”, “w” );
    while ( (ch = getche( ) ) != ‘\r’ )
        putc (ch, ptr);
    fclose (ptr);
}
```

- **FILE *fopen (const char *filename, const char *mode);**
  - modos
    - » r abre para leitura apenas.
    - » w cria para escrita. Se o arquivo existir será sobrescrito.
    - » a adicionar; abre para escrita no final do arquivo; cria um novo caso não exista.
    - » adicionando ‘**+**’ implica em poder também atualizar.
    - » **t** indica texto; **b** binário

Escrevendo Caractere

- int putc (int c, FILE *stream);
  - grava um caractere no arquivo
  - valor retornado
    - » se sucesso, retorna o caractere gravado
    - » caso contrário, EOF

Fechando o Arquivo

- int fclose (FILE *stream);
  - fclose fecha o arquivo
  - todos os **buffers** são descarregados
  - todas as áreas de comunicação alocados pelo sistema são liberados (**FILE**, **buffers**)
  - valor de retorno
    - » 0 se sucesso, EOF caso contrário

Lendo Arquivos

```
#include  < stdio.h >
main( ) {
    FILE  *ptr;    int  ch;

    ptr  =  fopen( “arqtext.txt”, “r” );
    while ( (ch = getc ( ptr ) ) != EOF )
        putchar (ch);
    fclose (ptr);
}
```

- int getc (FILE *stream);
  - complemento de putc
  - lê **um** caractere do arquivo
  - retorna o próximo caractere, incrementando o apontador (aponta para próximo caractere)
  - valor retornado
    - » se sucesso retorna o caractere lido
    - » EOF, caso contrário

Contando Caracteres

```
int  main (void)
{
    FILE  *ptr;       char  Path[30];     int  conta = 0;

    puts (“Digite o nome do arquivo:   ”);
    gets (Path);
    ptr = fopen(Path, "r");
    while ( getc (ptr) != EOF )  conta++;
    fclose (ptr);
    printf (“\n O arquivo tem %d caracteres”, conta);
}
```

Por que o número de carretares é menor do que o indicado pelo DOS?

Problemas

- os programas apresentados estão sujeitos a falhas

O que acontece se o arquivo não pode ser aberto?

é importante assegurar a correta abertura antes de qualquer operação que acesse um arquivo

- Solução
  - testar o resultado de fopen
  - retorna NULL se o arquivo não pode ser aberto

```
....
if  ((ptr = fopen( “arqtext.txt”, “r”))  ==  NULL) {
      printf (“ Impossível abrir arquivo “);
      exit( );
}
while ( (ch = getc (ptr) ) != EOF )
putchar (ch);
...
```

Gravando linha a linha

```
int  main (void)
{
    FILE  *ptr;     char  string[81];

    ptr = fopen("arqtext.txt", "w");
    while ( strlen ( gets (string) )  >  0 )  {
          fputs (string,  ptr);
          fputs ( "\n", ptr );
    }
    fclose (ptr);
}
```

- int fputs (const char *s, FILE *stream);
  - copia string terminado em null para um arquivo
  - não acrescenta caractere de nova linha
  - não copia '\0'
  - valor de retorno
    - » se sucesso retorna um valor não negativo
    - » EOF, caso contrário

Lendo linha a linha

```
int  main (void)
{
    FILE  *ptr;     char  string[81];

    ptr = fopen("arqtext.txt", "r");
    while ( fgets (string, 80, ptr)  !=  NULL)
          puts (string);
    fclose (ptr);
}
```

- char *fgets(char *s, int n, FILE *stream);
  - lê caracteres do arquivo para o string s
  - para após a leitura de n - 1 caracteres ou do caractere de nova linha (o que vier primeiro)
  - fgets retém o valor do caracter de nova linha
  - adicionando um byte (caractere) null marcando o final do string
  - valor de retorno
    - » se sucesso retorna um ponteiro para s
    - » NULL, caso contrário

Texto x Binário

- caractere ~ 1 byte
- números
  - int ~ 2 bytes, float ~ 4 bytes e etc
- modo texto
  - um número é gravado como um string
  - 1 204 347.20
    - » 4 bytes de memória principal
    - » 10 bytes em disco
  - ineficiente se lidamos com números e dados estruturados
- modo binário
  - incompatibilidade entre C e Ms-Dos
  - difere na interpretação de nova linha e na forma de representação de fim de arquivo

Nova Linha

- modo texto
  - caractere de nova linha transformado em CR/LF antes de ser gravado
  - CR/LF transformado em nova linha quando lido em C
- modo binário
  - nenhuma transformação é efetuada

```
int  main (void)
{
    FILE  *ptr;       int  conta = 0;

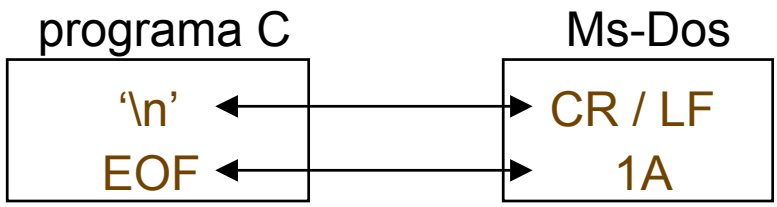
    ptr = fopen("teste.txt", "rb");
    while ( getc (ptr) != EOF )  conta++;
    fclose (ptr);
    printf ("\n O arquivo tem %d caracteres", conta);
}
```

- número de caracteres = indicado pelo S.O.

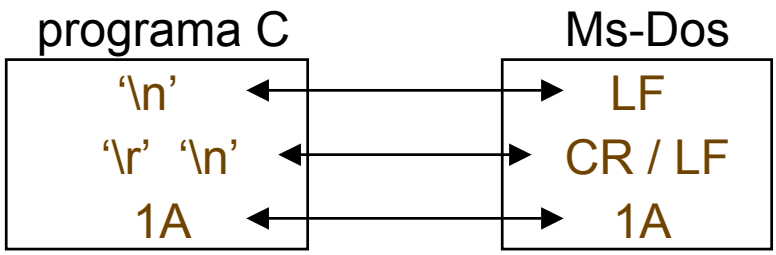
Fim de Arquivo

- modo texto
  - caractere especial **1A** (26 decimal) é inserido depois do último caractere
  - se este caractere for encontrado a função de leitura retorna EOF
- modo binário
  - caractere **1A** é lido como um caractere comum
  - LF (10 decimal) é gravado como LF e não como LF/CR (10 seguido de 13)

modo texto



modo binário



Leitura / Gravação de registros

- considere
  - struct dado {
    char nome [30];
    int idade;
    float nota; }
- como gravar os dados
  - como uma seqüência de strings (ineficiente)
  - um arquivo de registros

size_t fwrite(const void *p, size_t m, size_t n, FILE *f)

  - adiciona **n** itens de dados de tamanho **m** bytes ao arquivo indicado por **f**
  - **p** indica um apontador para **qualquer objeto** a ser gravado (posição inicial da memória)
  - número de bytes escritos = **(n * m)**
  - valor retornado
    - » se sucesso o **número de itens** gravados
    - » caso contrário, 0 (em geral)

Exemplo

```
typedef  struct   dado       {
                  int     codigo;
                  float  nota;    };

main( )   {
    dado  aluno =  { 0,  0 };       FILE * ptr;

    ptr = fopen("c:\\aluno", "wb");
    /*   cria um arquivo com 40 alunos  */
    do {
      fwrite (&aluno, sizeof(dado), 1, ptr);
      aluno.codigo++;
    } while  ( aluno.codigo < 40 )
  fclose(ptr);
}
```

- **CUIDADO**
  - fwrite (&aluno, sizeof (dado), 2, ptr)
    - » não grava o aluno 2 vezes

Leitura / Gravação de Registros

```
main( )
 {
    dado  aluno;       FILE * ptr;

    ptr = fopen("c:\\aluno", "rb");
    while  ( fread (&aluno, sizeof(dado), 1, ptr) )  {
         printf (“código =  %d “, aluno.codigo);
         printf (“nota  =  %f  \n“, aluno.nota);
    }
    fclose(ptr);
}
```

size_t fread (void ***p**, size_t **m**, size_t **n**, FILE ***f**)

- lê **n** itens de dado de tamanho **m** bytes para o bloco apontado por **p**
- o número de bytes lidos é **(n * m)**
- valor de retornado
  - » se sucesso, o número de itens lidos
  - » 0 caso contrário

Gravando Arranjos

```
main( )
 {
    int  numeros[40];       FILE * ptr;

    for ( i = 0;  i  <  40; i++ )  numeros[ i ] = i;
    ptr = fopen("num.dat", "wb");
    fwrite ( &numeros, sizeof ( int ), 40, ptr );
    fclose(ptr);
}
```

fwrite(&numeros, sizeof (int), 40, ptr)

=

fwrite(&numeros, sizeof (numeros), 1, ptr)

mas

fwrite(&numeros, sizeof (int), 30, ptr)

» grava apenas as 30 posições iniciais

Lendo Arranjos

```
main( )
 {
    int  numeros[40];       FILE * ptr;

    for ( i = 0;  i  <  40; i++ )  numeros[ i ] = 0;
    ptr = fopen("num.dat", "rb");
    fread ( &numeros, sizeof ( int ), 40, ptr );
    fclose(ptr);
}
```

fread (&numeros, sizeof (int), 40, ptr)

=

fread (&numeros, sizeof (numeros), 1, ptr)

mas

fread(&numeros, sizeof (int), 30, ptr)

- lê apenas as 30 posições iniciais
- resto continua inalterado (zero)

Acesso Aleatório

- permite acessar uma posição do arquivo independente da demais
- considere um arquivo de alunos

```
main( ) {
    dado  aluno;      FILE * ptr;
    int   num_reg;    long int   offset;

    ptr = fopen("c:\\aluno", "rb");

    printf (“Qual registro deseja ler  < 1 a 40 > ”);
    scanf (“%d”, &num_reg);

    offset = (num_reg - 1) * sizeof ( dado );
    fseek ( ptr, offset, 0 );
    fread (&aluno, sizeof (dado), 1, ptr) ;

    printf (“código =  %d “, aluno.codigo);
    printf (“nota  =  %f  \n“, aluno.nota);
    fclose(ptr);
}
```

Funções para Acesso Aleatório

- int fseek (FILE ***f**, long **offset**, int **apartir**);
  - reposiciona o apontador de um arquivo **offset** bytes da localização dada por **apartir**
  - para modo texto **apartir** deve ser **0**
  - **apartir** deve assumir um dos valores 0, 1, ou 2 representado pelas constantes (stdio.h)
    - » SEEK_SET 0 começo do arquivo
    - » SEEK_CUR 1 posição corrente
    - » SEEK_END 2 fim do arquivo
  - valor de retorno
    - » se sucesso 0
    - » caso contrário, não-zero

A Função ftell( )

- long int ftell (FILE *f)
  - retorna o offset em bytes calculado a partir do início do arquivo (binário)
  - valor retornado
    - » se sucesso a posição corrente
    - » caso contrário, -1L

```
long   tamanho_arquivo (FILE *arquivo)
{
  long   posicao,  tamanho;

  posicao  =  ftell (arquivo);
  fseek (arquivo, 0L, SEEK_END);
  tamanho = ftell ( arquivo );
  fseek ( arquivo, posicao, SEEK_SET);
  return tamanho;
}
```