UNIDADE IV: ARQUIVOS

Conceitos Básicos

Conjuntos de informações mantidas no disco (memória secundária)

Informações são **"persistidas"** em comparação com a memória RAM (memória primária)

Sistema Operacional (OS) faz um "buffer" das informações lidas/gravadas

Conceitos Básicos

Em C, arquivos podem ser gravados de duas maneiras:

Modo texto (conjunto de caracteres)

Arquivo texto pode ser tratado por qualquer editor (e.g., bloco de notas, terminal, word, etc.)

Modo binário (conjunto de bytes)

Ex: Grandes quantidades de informações de forma eficiente

Operações Primordiais

- **Abertura do arquivo** (localização, alocação do buffer)
- **Leitura do arquivo** (informações do buffer são disponibilizadas)
- **Gravação do arquivo** (alteração de dados preexistentes ou acréscimo de novos dados)
- **Fechamento do arquivo** (atualização do buffer e liberação de memória alocada)

Comandos Essenciais

```
FILE *fp;

FILE *fopen(char *nome_arquivo,  char
*modo_de_acesso);

int fclose(FILE *fluxo);
```

Possíveis Modos de Acesso

modo_de_acesso	Significado
r	Abre o arquivo somente para leitura. O arquivo deve existir.
r+	Abre o arquivo para leitura e escrita. O arquivo deve existir.
w	Abre o arquivo somente para escrita no início do arquivo. Apagará o conteúdo do arquivo se ele já existir, criará um arquivo novo se ele não existir.
w+	Abre o arquivo para escrita e leitura, apagando o conteúdo pré-existente.
a	Abre o arquivo para escrita no final do arquivo. Não apagará o conteúdo pré-existente.
a+	Abre o arquivo para escrita no final do arquivo e leitura.

Ex.: Criando um Arquivo

```
#include <stdio.h>

int main()
{
    FILE *fp;
    fp = fopen ("README", "w");
    if (fp == NULL) {
       printf ("Houve um erro ao abrir o arquivo.\n");
       return 1;
    }
    printf ("Arquivo README criado com sucesso.\n");
    fclose (fp);
    return 0;
}
```

Ex.: Lendo um Arquivo

```
#include <stdio.h>

int main()
{
    char c[41];
    FILE *fp;

    fp = fopen("README", "r");
    while(fgets(c, 40, fp) != NULL)
       puts(c);
    fclose(fp);
    return 0;
}
```

Ex.: Contando linhas de um Arquivo

```
#include <stdio.h>

int main()
{
    int c, nlinhas = 0;
    FILE *fp;
    fp = fopen("README", "r");

    while((c=fgetc(fp))!=EOF)
        if(c=='\n') nlinhas++;

    fclose(fp);
    printf("Numero de linhas = %d\n", nlinhas);
    return 0;
}
```

Escrevendo em um arquivo

```
void fputc(int caractere, FILE *fluxo)

void fputs(char *string, FILE *fluxo)

void fprintf(FILE *fluxo, char *formatação)

int  fwrite(void *dados, int
tamanho_do_elemento,  int num_elementos,
FILE *fluxo)
```

Escrevendo em um arquivo

Saída padrão	Arquivos	Explicação
putchar	fputc	Imprime apenas um caractere
puts	fputs	Imprime uma string diretamente, sem nenhuma formatação
printf	fprintf	Imprime uma string formatada
N/A	fwrite	Grava dados binários em um arquivo

Ex.: Escrevendo em um Arquivo

```c
#include <stdio.h>

int main()
{
    FILE *fp;
    fp = fopen ("README", "w");
    fprintf(fp, "Salvando alguma coisa no meu arquivo...\n");
    fclose(fp);

    return 0;
}
```

Ex.: Escrevendo em um Arquivo

```c
#include <stdio.h>

int main()
{
    FILE *fp;
    char frase[] = "Universidade Federal de Minas Gerais";
    fp = fopen("file", "w");

    for(int i = 0; frase[i]!='\0'; i++)
        fprintf(fp, "%c", frase[i]);

    return 0;
}
```

Arquivos Binários

- Servem para salvar (recuperar) as informações tais como se encontram na memória principal
- Permite o manuseio de grandes quantidades de dados de forma mais eficiente (i.e., não sequencial)
- Um arquivo binário permite a busca por qualquer informação contida no arquivo (*fseek*)

Arquivos Binários

```
int fread(void* p, int tam, int num,
FILE *fp)

int fwrite(void* p, int tam, int  num,
FILE *fp)

int fseek(FILE *fp, long offset, int
origem)
```

Arquivos Binários

```
#include <stdio.h>

int main()
{
   FILE *fp;  fp=fopen("test.bin", "wb");
   int x[5]= {1, 2, 3, 4, 5};

   fwrite(x, sizeof(x[0]), sizeof(x)/sizeof(x[0]), fp);

   return 0;
}
```


The End