Aula 10 – Manipulação de Arquivos

1. Introdução

A linguagem C permite a manipulação de arquivos para leitura e impressão, da mesma forma que temos feito leitura do teclado e impressão em tela. Nesta aula aprenderemos como.

2. Fluxos

- Entrada e saída em C é feita por meio de fluxos: `scanf()` e `printf()` fazem, respectivamente, leitura e escrita no fluxo padrão, o console (modo texto) do sistema operacional;
- É possível também utilizar *strings* como fluxos com `sscanf()` e `sprintf()`:

```
#include <stdio.h>
#include <string.h>

main() {
     char string[100];
     float f;

     // Escreve um número numa string:
     f = 3.141592;
     sprintf(string, "f = %1.1f!", f);
     printf("%s\n", string); // f = 3.1!

     // Converte de string para número:
     strcpy(string, "3.141592");
     sscanf(string, "%f", &f);
     printf("%1.4f\n", f); // 3.1416
}
```

- Como pode ser visto, `sscanf()` e `sprintf()` podem ser utilizadas para conversão entre *string* e tipos numéricos (inteiros e reais).

3. Fluxos para arquivos

- Leitura e escrita em arquivos não é diferente: é feita por meio de fluxos, utilizando as funções `fscanf()` e `fprintf()`;
- Um fluxo de/para arquivo é representado pelo tipo (não primitivo) `FILE`, definido em `<stdio.h>`. Em nossos códigos, utilizamos sempre um ponteiro para `FILE`: `FILE *arquivo`;
- No caso de leitura de *strings*, `fscanf()` se comporta da mesma maneira que `scanf()`, lendo pedaços de *string* de cada vez, separados por espaço. Para ler uma linha inteira, da mesma forma que usamos `gets()` ao invés de `scanf()`, podemos usar `fgets()` ao invés de `fscanf()`.

3.1. Leitura e escrita textual

As seguintes funções nos permitem fazer leitura de texto em arquivo:

- `FILE *fopen(char *nomeArquivo, char *modo)`: abre um fluxo para arquivo;
    - `fopen()` retorna NULL se ocorreu algum erro;

- O argumento `modo` especifica o tipo de operação que se deseja realizar no arquivo que está sendo aberto:

Modo	Resultado
`"r"`	Fluxo somente leitura. O arquivo deve já existir.
`"w"`	Fluxo somente escrita. O arquivo é criado ou sobrescrito.
`"a"`	Fluxo somente escrita. Se o arquivo já existe, adiciona ao final. Do contrário, cria um novo arquivo.
`"r+"`	Fluxo para leitura e gravação. O arquivo deve já existir.
`"w+"`	Fluxo para leitura e gravação. O arquivo é criado ou sobrescrito.
`"a+"`	Fluxo para leitura e gravação. Se o arquivo já existe, adiciona ao final. Do contrário, cria um novo arquivo.

- `int fclose(FILE *arquivo)`: fecha um fluxo para arquivo;
  - Por questões de eficiência, os dados escritos são armazenados num *buffer* de memória e levados ao arquivo somente de tempos em tempos;
  - Fechar um arquivo escreve o restante do buffer no arquivo. Se não o fizermos, podemos perder dados!
  - A função retorna 0 (zero) se tudo ocorreu sem erros.

- Como já mencionado, a leitura e escrita em si é feita utilizando funções da família printf/scanf:
  - `int fprintf(FILE *arquivo, char *formato, ...);`
  - `int fscanf(FILE *arquivo, char *formato, ...);`

- `int feof(FILE *arquivo)`: para saber se chegamos ao final do arquivo:
  - `feof()` retornará `0` (falso) se o arquivo ainda não chegou ao final e `1` (verdadeiro) se já chegou;
  - Deve ser utilizado em combinação com `fscanf()` exclusivamente (veremos como ler um arquivo todo com `fgets()` posteriormente);
  - Atenção: `feof()` só começará a retornar 1 após uma tentativa de utilizar `fscanf()` sem nenhum dado para ser lido. Ou seja, após ler a última informação no arquivo, `feof()` ainda retornará 0. Somente depois de tentar ler algo que não existe é que ele passa a retornar 1!

Exemplo – leitura de um arquivo, item por item (abaixo: *string*; mas poderia ser inteiro, real, etc.):

```
FILE *arquivo;
char palavra[50];
arquivo = fopen("exemplo.txt", "r");
fscanf(arquivo, "%s", palavra);        // Não lê linhas em branco.
while (! feof(arquivo)) {
     printf("Leu: %s\n", palavra);
     fscanf(arquivo, "%s", palavra);
}
fclose(arquivo);
```

- `char *fgets(char *string, int tamanho, FILE *arquivo)`: lê até o final da linha ou até o tamanho especificado como segundo parâmetro:
  - `fgets()` retorna um ponteiro para a *string* lida ou NULL caso não seja possível ler mais nada do arquivo. Para ler um arquivo inteiro, portanto, verificamos se o retorno da função é nulo (ver exemplo a seguir);
  - Se uma linha inteira for lida, `fgets()` incluirá o caractere de quebra de linha '\n' na *string*. Para removê-lo, substituímo-lo por '\0' (caractere de fim de *string*). Ver exemplo a seguir.

Exemplo – leitura de um arquivo, linha por linha:

```
FILE *arquivo;
char linha[1000];
arquivo = fopen("exemplo.txt", "r");
while (fgets(linha, sizeof(linha), arquivo)) {     // Lê linhas em
     linha[strlen(linha) - 1] = '\0';              // branco.
     printf("Leu: %s\n", linha);
}
fclose(arquivo);
```

3.2. Leitura e escrita binária

Para não precisar converter sempre dados para *string* e vice-versa, é possível armazená-los em arquivos diretamente em forma binária, utilizando um dos seguintes modos na função `fopen()`:

Modo	Resultado
`"rb"`	Fluxo binário somente leitura, funciona como o `"r"`.
`"wb"`	Fluxo binário somente escrita, funciona como o `"w"`.
`"ab"`	Fluxo binário somente escrita, funciona como o `"a"`.
`"r+b"`	Fluxo binário para leitura e gravação, funciona como o `"r+"`.
`"w+b"`	Fluxo binário para leitura e gravação, funciona como o `"w+"`.
`"a+b"`	Fluxo binário para leitura e gravação, funciona como o `"a+"`.

Além disso, a leitura/escrita deve ser feita com funções diferentes (não é com fscanf/fprintf):

- `fread(void *buffer, int tamanho, int qtd, FILE *arquivo);`
  - `buffer`: onde serão armazenados os dados lidos;
  - `tamanho`: tamanho da estrutura a ser lida;
  - `qtd`: quantidade de itens (todos da mesma estrutura) a serem lidos;
  - `arquivo`: fluxo de arquivo de onde será feita a leitura.

- `fwrite(void *buffer, int tamanho, int qtd, FILE *arquivo);`
  - Idem `fread()`, porém para escrita.
  - Ambos retornam 1 se ocorreu tudo sem erros.

Exemplo:

```
FILE *pf;
```

```
float pi = 3.1415, pi2;

pf = fopen("arq.bin", "wb");
if(fwrite(&pi, sizeof(float), 1, pf) != 1)
     printf("Erro na escrita do arquivo");
fclose(pf);

pf = fopen("arq.bin", "rb");
if(fread(&pi2, sizeof(float), 1, pf) != 1)
     printf("Erro na leitura do arquivo");
printf("pi2 = %f\n", pi2);
fclose(pf);
```

4. Fluxos pré-definidos

- `stdin` (*standard input):* fluxo padrão de entrada (console/teclado);
- `stdout` (*standard output*): fluxo padrão de saída (console/tela);
- `stderr` (*standard error*): fluxo padrão de saída de erros (console/tela);
- No Linux, é possível direcionar (1 = saída normal, 2 = saída de erro):

```
./programa 1>padrao.txt 2>erro.txt
```