



Lendo arquivos com fscanf()

O que é fscanf ?

`fscanf(FILE *arquivo, const char *formato, ...)` funciona de forma parecida com `scanf`, mas lê **dados formatados diretamente de um arquivo**.

- Ele tenta **casar o conteúdo do arquivo com o formato especificado**.
- Cada item do formato (`%d`, `%s`, `%c`, etc.) espera um **ponteiro para a variável** onde vai guardar o valor lido.

Sintaxe Básica:

```
FILE *arquivo = fopen("dados.txt", "r");
int idade;
char nome[100];

fscanf(arquivo, "%s %d", nome, &idade);
```

- `%s`: lê uma string até espaço (sem espaços internos).
- `%d`: lê um número inteiro.
- `%c`: lê **um caractere** (inclusive espaços).

- `%f` : lê números decimais (float/double).



Exemplos práticos de formatos de arquivo .txt + como ler com fscanf

1. Arquivo com: nome;idade;bairro

Exemplo de `arquivo1.txt` :

```
João;25;Centro
Maria;30;Jardin
```

Como ler:

```
char nome[50], bairro[50];
int idade;

FILE *arq = fopen("arquivo1.txt", "r");

while (fscanf(arq, "%[^;];%d;%[^\\n]\\n", nome, &idade, bairro) == 3) {
    printf("Nome: %s, Idade: %d, Bairro: %s\\n", nome, idade, bairro);
}

fclose(arq);
```

`%[^;]` → lê até encontrar

`%[^\\n]` → lê até a quebra de linha (final da linha)

1) Uso de == n

Por que usamos

`== 3` em

```
while (fscanf(arq, "%[^;];%d;%[^\\n]\\n", nome, &idade, bairro) == 3)
```

`fscanf` retorna o número de itens que foram lidos com sucesso.

- `%[^;]` → espera ler a **string** até `;` → 1 item
- `%d` → espera ler a **idade (inteiro)** → +1 item
- `%[^\\n]` → espera ler a **string até o fim da linha** (bairro) → +1 item

Total esperado: 3 itens lidos com sucesso.

2) Scan set

O que são

`%[]` e `%[^]`?

- `%[abc]` → leia e armazene todos os caracteres **enquanto forem 'a', 'b' ou 'c'**.
- `%[^xyz]` → leia e armazene todos os caracteres **enquanto NÃO forem 'x', 'y' ou 'z'**.

Exemplos rápidos:

Formato	Significado
<code>%[^,]</code>	Lê até encontrar vírgula (,)
<code>%[^;]</code>	Lê até ponto e vírgula (;)
<code>%[A-Za-z]</code>	Lê apenas letras
<code>%[0-9]</code>	Lê apenas números
<code>%[^ \\n]</code>	Lê até espaço ou quebra de linha

2. Arquivo com: uma

frase por linha (string com espaços)

```
Hoje o tempo está bom.  
Amanhã pode chover.
```

```
char linha[200];  
  
FILE *arq = fopen("frases.txt", "r");  
  
while (fgets(linha, sizeof(linha), arq) != NULL) {  
    printf("Frase: %s", linha);  
}  
  
fclose(arq);
```



`fscanf` não é bom para strings com espaços. Use `fgets` aqui.

✓ 3. Arquivo com: números inteiros separados por espaço

Exemplo de `numeros.txt`:

```
Copiar código  
10 20 30 40 50
```

Como ler:

```
int num;  
  
FILE *arq = fopen("numeros.txt", "r");  
  
while (fscanf(arq, "%d", &num) == 1) {  
    printf("Número: %d\n", num);  
}
```

```
fclose(arq);
```

✓ 4. Arquivo com: nome e notas separados por vírgula

Exemplo de **alunos.txt**

```
Carlos,8.5,7.0,9.0  
Fernanda,10.0,9.5,8.6
```

Como ler:

```
char nome[50];  
float n1, n2, n3;  
  
FILE *arq = fopen("alunos.txt", "r");  
  
while (fscanf(arq, "%[^,],%f,%f,%f", nome, &n1, &n2, &n3) == 4) {  
    printf("Aluno: %s - Médias: %.2f, %.2f, %.2f\n", nome, n1, n2, n3);  
}  
  
fclose(arq);
```

✓ 5. Arquivo com: data no formato **dd/mm/aaaa**

Exemplo de **datas.txt** :

```
13/05/2025  
01/01/2025
```

Como ler:

```
int dia, mes, ano;

FILE *arq = fopen("datas.txt", "r");

while (fscanf(arq, "%d/%d/%d", &dia, &mes, &ano) == 3) {
    printf("Data: %02d-%02d-%04d\n", dia, mes, ano);
}

fclose(arq);
```

✓ 6. Arquivo com apenas caracteres (como em criptografia)

Exemplo de **mensagem.txt** :

```
Hello, world
```

Como ler caractere por caractere:

```
char c;

FILE *arq = fopen("mensagem.txt", "r");

while (fscanf(arq, "%c", &c) == 1) {
    // Aqui você pode aplicar criptografia
    printf("%c", c);
}

fclose(arq);
```

✓ 7. Arquivo com: booleanos ou letras específicas

Exemplo de **respostas.txt** :

```
S
N
```

S

Como ler:

```
char resposta;  
  
FILE *arq = fopen("respostas.txt", "r");  
  
while (fscanf(arq, " %c", &resposta) == 1) {  
    printf("Resposta: %c\n", resposta);  
}  
  
fclose(arq);
```

 O espaço antes de `%c` é importante para **ignorar quebras de linha e espaços.**