

SSC0501 - Introdução à Ciência de Computação I

Bem Vindos!

Arquivos Continuação!

Arquivos Binários

Arquivos Binários

- Arquivos binários armazenam dados em seu formato bruto, sem conversão para texto.
- **São mais eficientes em espaço e tempo de acesso.**
- Não podem ser lidos diretamente por editores de texto.

Exemplos de arquivos binários: imagens (.jpg, .png), executáveis (.exe), banco de dados, etc.

Diferença entre Arquivo Texto e Arquivo Binário

Arquivo Texto	Arquivo Binário
Armazena dados como caracteres ASCII	Armazena dados no formato bruto (bytes)
Pode ser lido por editores de texto	Requer programas específicos para leitura
Uso de funções como fprintf, fscanf	Uso de fwrite, fread

Abrindo Arquivos Binários: fopen()

```
1#include <stdio.h>
2
3int main()
4{
5    FILE *fp;
6    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
7
8    if (fp == NULL)
9    {
10        printf("Erro ao abrir o arquivo.\n");
11        return 1;
12    }
13
14    fclose(fp);
15
16    return 0;
17}
```

[abrindo_arquivo_binario.c](#)

Abrindo Arquivos Binários: **fopen()**

```
1 #include <stdio.h>
2
3 int main()
4 {
5     FILE *fp;
6     fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
7
8     if (fp == NULL)
9     {
10         printf("Erro ao abrir o arquivo.\n");
11         return 1;
12     }
13
14     fclose(fp);
15
16     return 0;
17 }
```

Acrescenta o "b" para modo binário

[abrindo_arquivo_binario.c](#)

Arquivos Binários - Modos de Abertura

- **rb** - leitura binária
(abre o arquivo para leitura; falha se o arquivo não existir)
- **wb** - escrita binária
(cria um novo arquivo ou apaga o conteúdo existente)
- **ab** - escrita binária no final (anexar)
(cria o arquivo, se não existir; grava sempre no final)

Escrita de Dados com **fwrite()**

Escrita de Dados com **fwrite()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26}
```

[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26}
```

[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26}
```

Escreve a struct criada no arquivo em binário

[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26}
```

Qual o tamanho do arquivo criado?

Escreve a struct criada no arquivo em binário

[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id; 4 bytes
5     float nota;
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

Qual o tamanho do arquivo criado?

Escreve a struct criada no arquivo em binário

[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id; 4 bytes
5     float nota; 4 bytes
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

Qual o tamanho do arquivo criado?

Escreve a struct criada no arquivo em binário

[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id; 4 bytes
5     float nota; 4 bytes
6 } Aluno; Total = 8 bytes = sizeof(Aluno)
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

Qual o tamanho do arquivo criado?

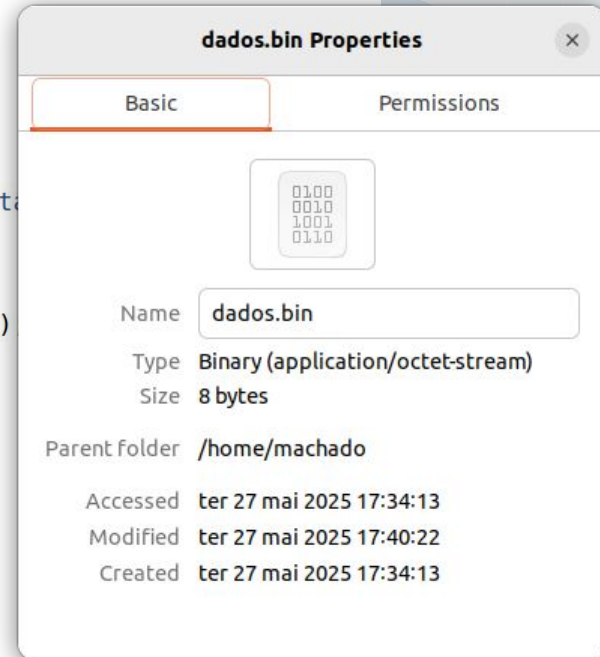
Escreve a struct criada no arquivo em binário

[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id; 4 bytes
5     float nota; 4 bytes
6 } Aluno; Total = 8 bytes = sizeof(Aluno)
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

Qual o tamanho do arquivo criado?

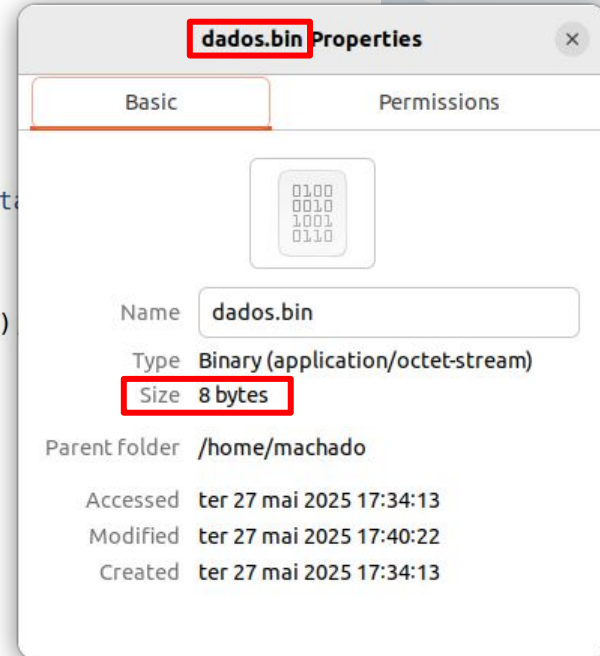


[escrevendo_arquivo_binario.c](#)

Escrita de Dados com **fwrite()**

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id; 4 bytes
5     float nota; 4 bytes
6 } Aluno; Total = 8 bytes = sizeof(Aluno)
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

Qual o tamanho do arquivo criado?



[escrevendo_arquivo_binario.c](#)

Como visualizar um arquivo binário?

Como visualizar um arquivo binário?

1 - Programas específicos como visualizador de imagens para imagens, player de vídeo para vídeos, visualizador de PDF para PDF, etc...

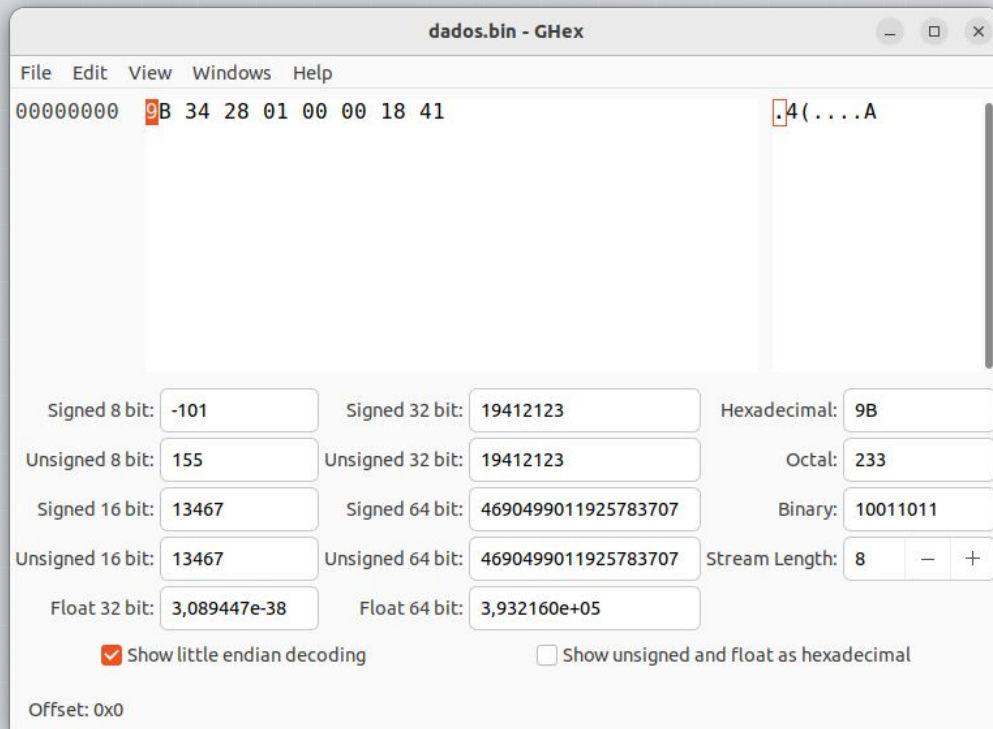
2 - Visualizadores/Editores de arquivos binários. Exemplo: **gHex**

Instalação no Linux:

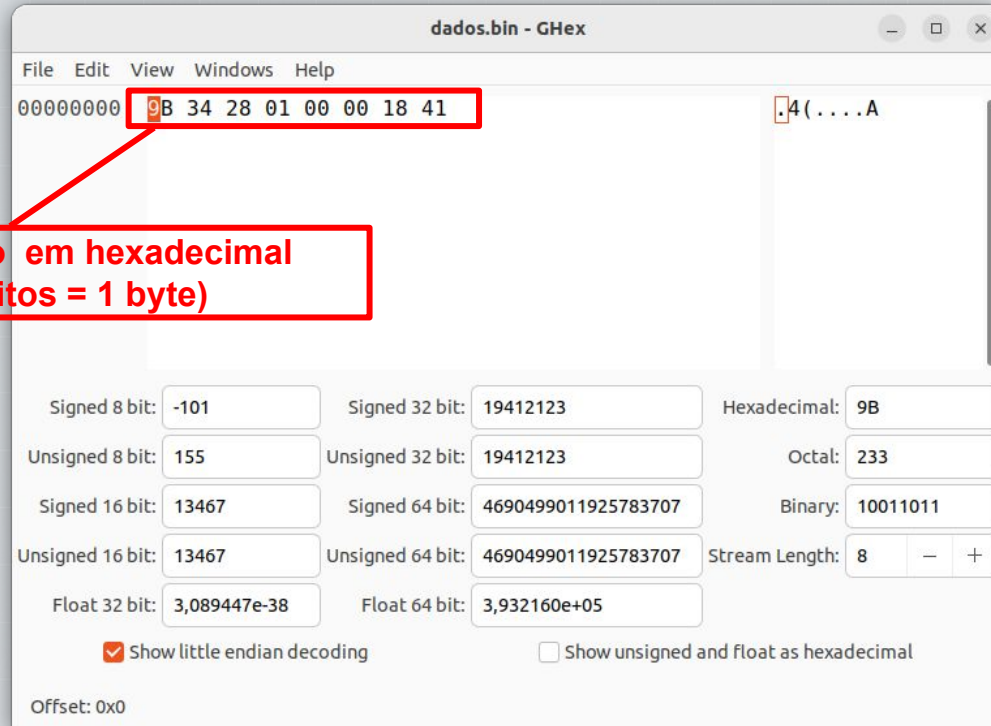
```
sudo apt install ghex
```

Como visualizar um arquivo binário?

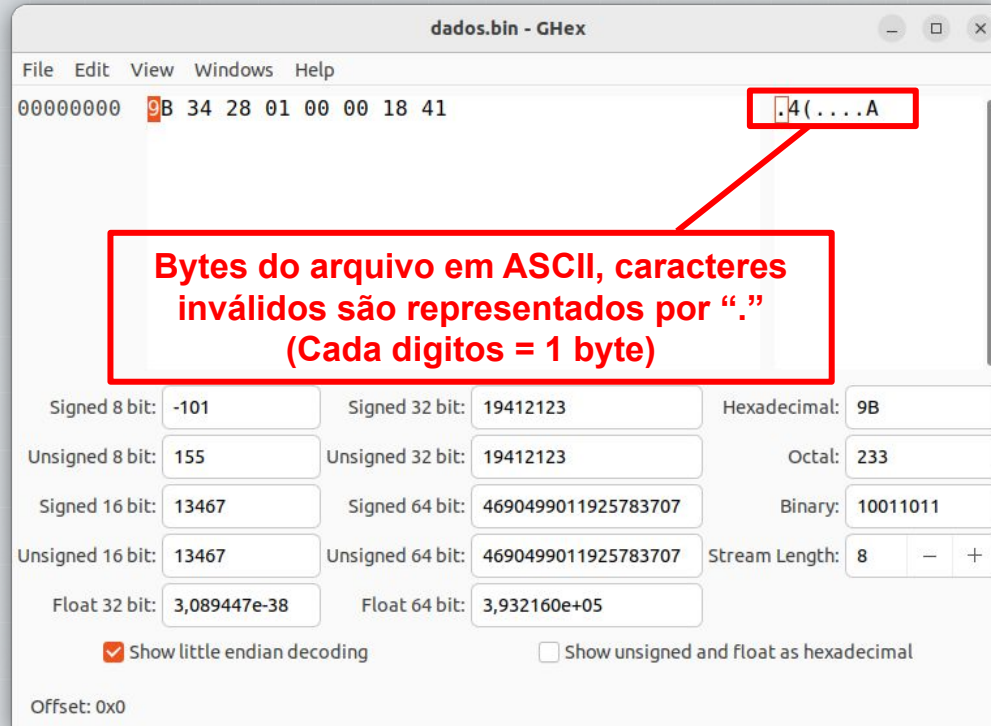
Abrindo o **dados.bin** (último exemplo) com o **GHex**



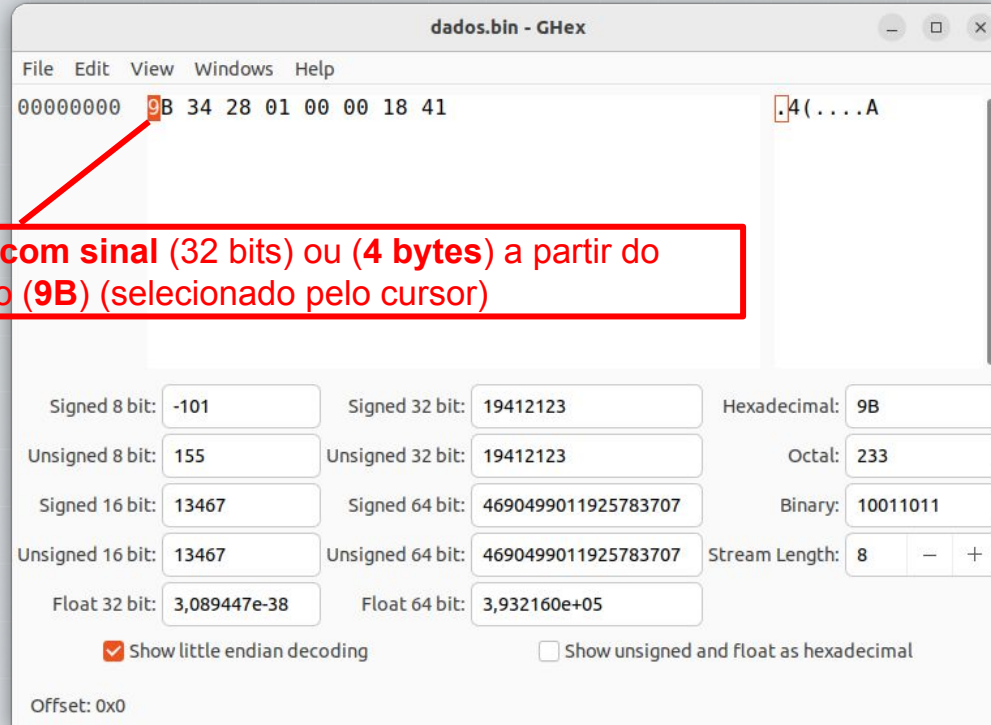
Como visualizar um arquivo binário?



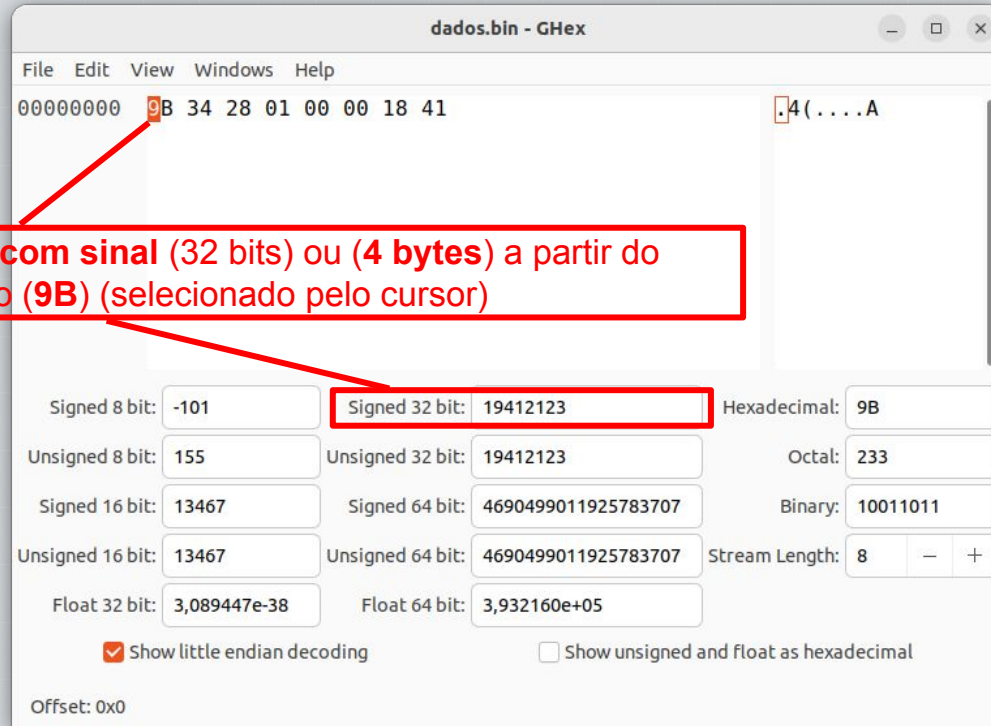
Como visualizar um arquivo binário?



Como visualizar um arquivo binário?

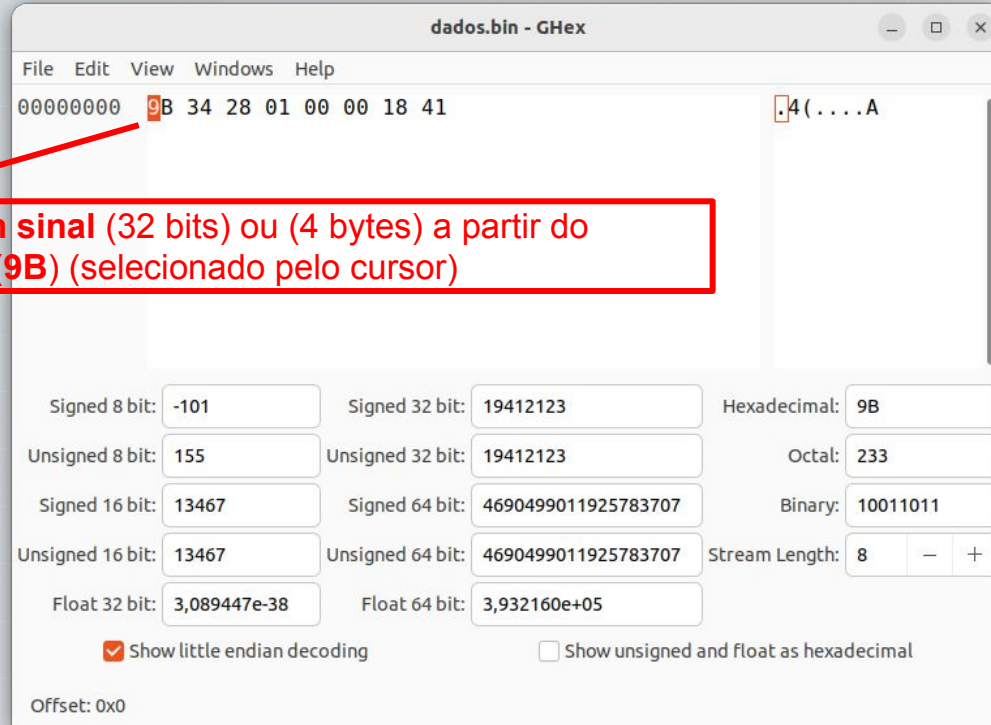


Como visualizar um arquivo binário?



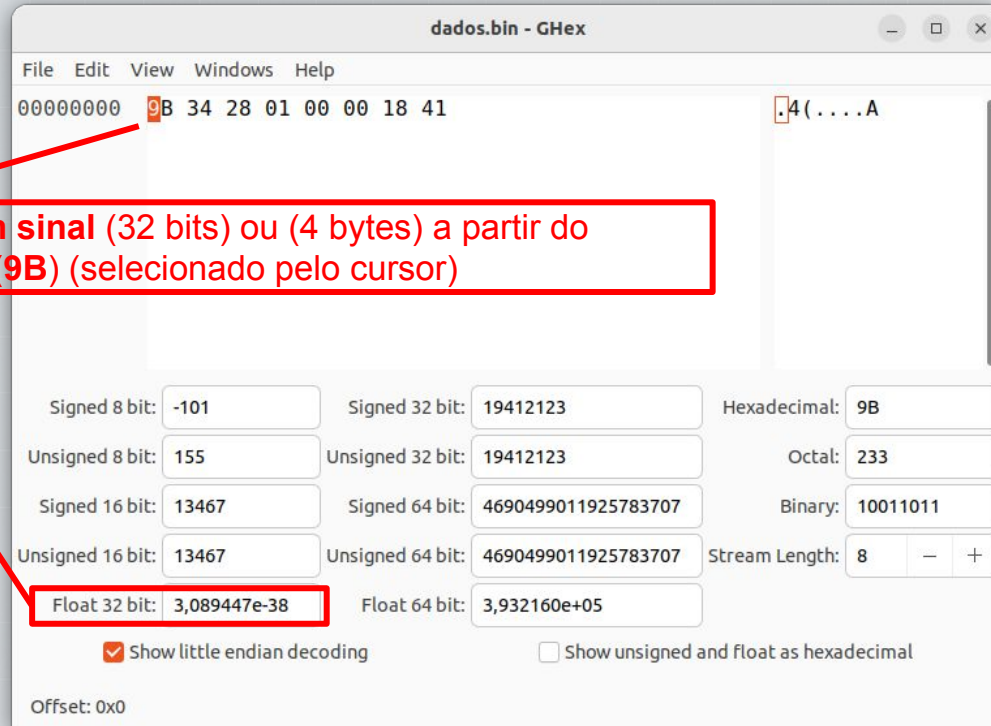
Como visualizar um arquivo binário?

Interpretação de **float com sinal** (32 bits) ou (4 bytes) a partir do **primeiro byte** do arquivo (**9B**) (seleccionado pelo cursor)

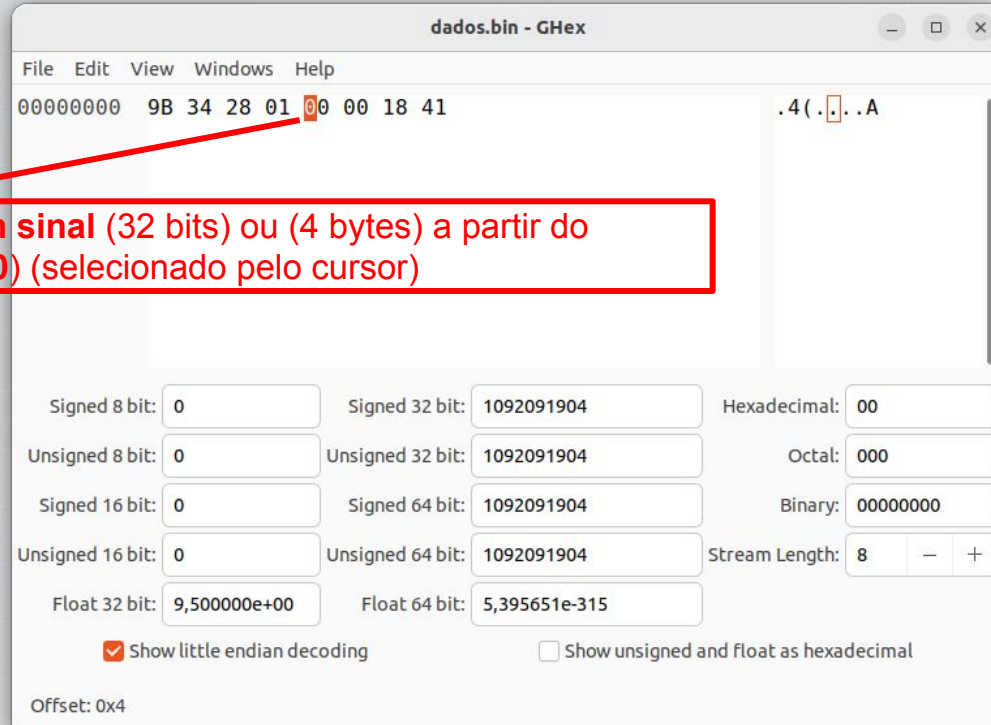


Como visualizar um arquivo binário?

Interpretação de **float com sinal** (32 bits) ou (4 bytes) a partir do **primeiro byte** do arquivo (**9B**) (seleccionado pelo cursor)



Como visualizar um arquivo binário?



Como visualizar um arquivo binário?

Interpretação de **float com sinal** (32 bits) ou (4 bytes) a partir do **quinto byte** do arquivo (**00**) (selecionado pelo cursor)

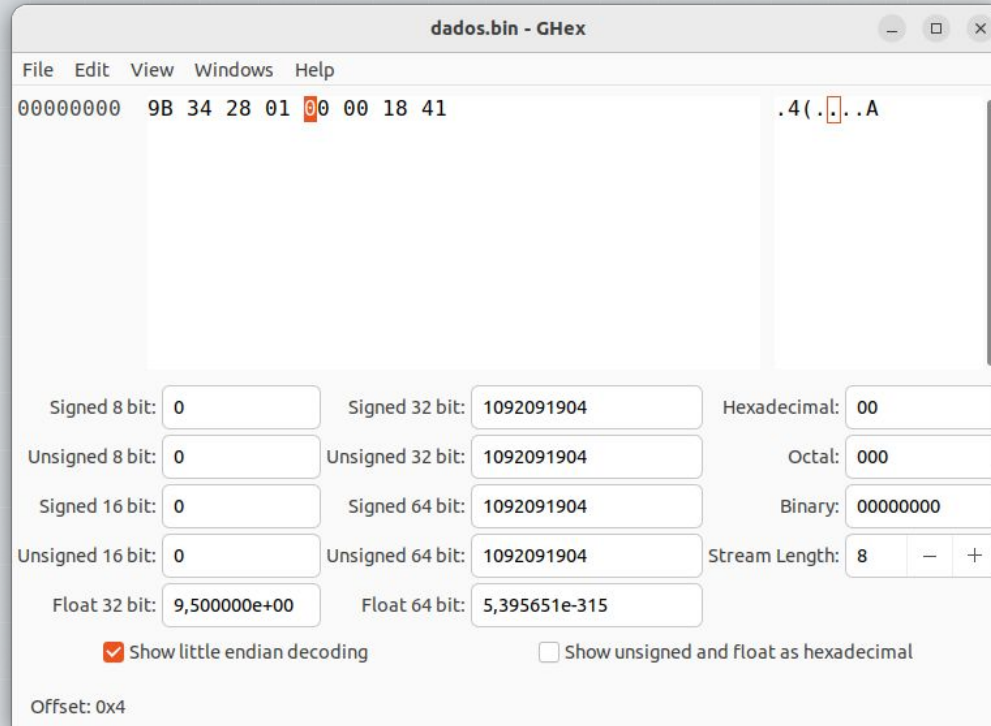
The screenshot shows the GHex application window titled "dados.bin - GHex". The menu bar includes File, Edit, View, Windows, and Help. The main pane displays the binary data "00000000 9B 34 28 01 00 00 18 41" in hexadecimal. The fifth byte, "00", is highlighted with a red box and a red arrow pointing to the interpretation section below. The interpretation section contains several input fields for different data types: Signed 8 bit: 0, Unsigned 8 bit: 0, Signed 16 bit: 0, Unsigned 16 bit: 0, Signed 32 bit: 1092091904, Unsigned 32 bit: 1092091904, Signed 64 bit: 1092091904, Unsigned 64 bit: 1092091904, Hexadecimal: 00, Octal: 000, Binary: 00000000, Stream Length: 8. The "Float 32 bit" field is highlighted with a red box and contains the value "9,500000e+00". Below the interpretation fields, there are two checkboxes: "Show little endian decoding" (checked) and "Show unsigned and float as hexadecimal" (unchecked). The "Offset: 0x4" is displayed at the bottom left.

Signed 8 bit:	0	Signed 32 bit:	1092091904	Hexadecimal:	00
Unsigned 8 bit:	0	Unsigned 32 bit:	1092091904	Octal:	000
Signed 16 bit:	0	Signed 64 bit:	1092091904	Binary:	00000000
Unsigned 16 bit:	0	Unsigned 64 bit:	1092091904	Stream Length:	8
Float 32 bit:	9,500000e+00	Float 64 bit:	5,395651e-315		

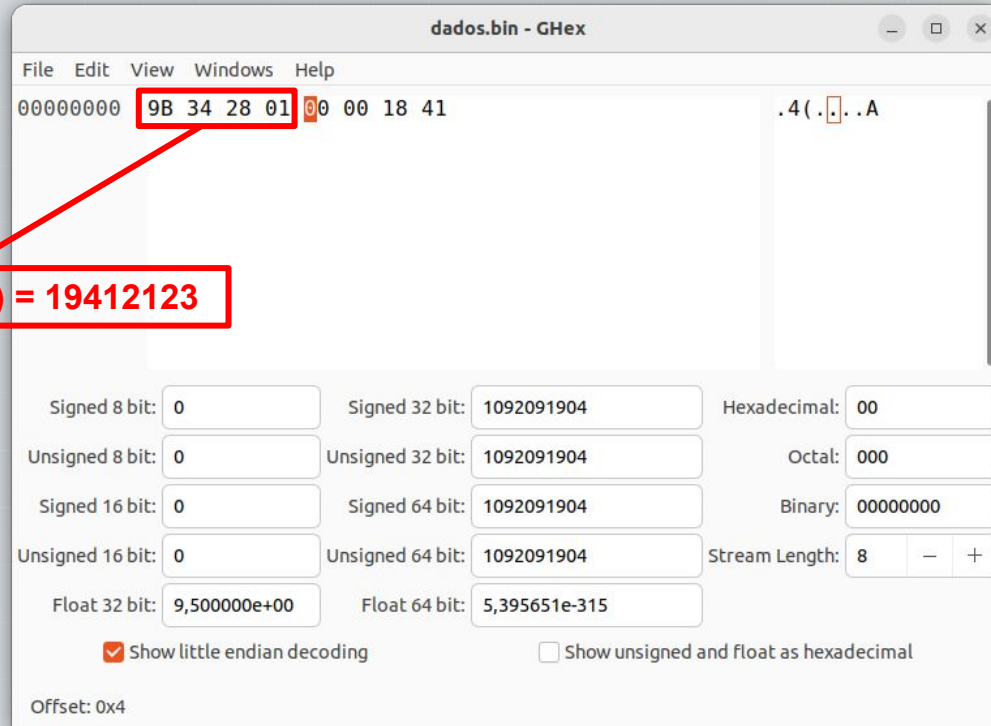
☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x4

Como visualizar um arquivo binário?



Como visualizar um arquivo binário?



Como visualizar um arquivo binário?

The screenshot shows the GHex application window titled "dados.bin - GHex". The menu bar includes File, Edit, View, Windows, and Help. The main display area shows the hex address 00000000 and the hex data 9B 34 28 01 00 00 18 41. The data is displayed in a hex editor view with a corresponding ASCII view on the right showing ".4 (. . . A".

Red boxes and arrows highlight specific parts of the data:

- A red box around the first four bytes (9B 34 28 01) is connected by a red arrow to a red box containing the text: **Inteiro (4 bytes) = 19412123**
- A red box around the next four bytes (00 00 18 41) is connected by a red arrow to a red box containing the text: **Float (4 bytes) = 9.5**

Below the hex editor, there are various conversion options and their results:

Conversion Type	Value
Signed 8 bit:	0
Unsigned 8 bit:	0
Signed 16 bit:	0
Unsigned 16 bit:	0
Float 32 bit:	9,500000e+00
Signed 32 bit:	1092091904
Unsigned 32 bit:	1092091904
Signed 64 bit:	1092091904
Unsigned 64 bit:	1092091904
Float 64 bit:	5,395651e-315
Hexadecimal:	00
Octal:	000
Binary:	00000000
Stream Length:	8

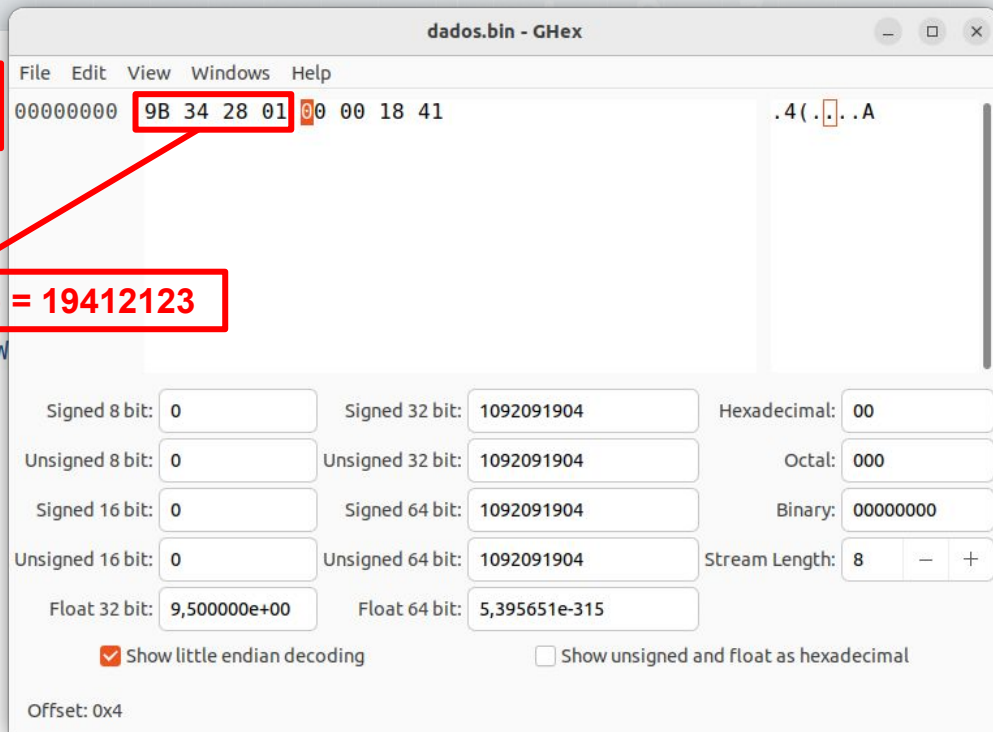
At the bottom, there are two checkboxes: ☒ Show little endian decoding and ☐ Show unsigned and float as hexadecimal. The offset is shown as 0x4.

Como visualizar um arquivo binário?

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id;
5     float nota;
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita binário (W
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

O TIPO do dado informa como devemos interpretar os BYTES!

Inteiro (4 bytes) = 19412123



Como visualizar um arquivo binário?

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (W
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26}
```

O TIPO do dado informa como devemos interpretar os BYTES!

Inteiro (4 bytes) = 19412123

dados.bin - GHex

File Edit View Windows Help

00000000 9B 34 28 01 00 00 18 41 .4(. .A

Signed 8 bit: 0 Signed 32 bit: 1092091904 Hexadecimal: 00

Unsigned 8 bit: 0 Unsigned 32 bit: 1092091904 Octal: 000

Signed 16 bit: 0 Signed 64 bit: 1092091904 Binary: 00000000

Unsigned 16 bit: 0 Unsigned 64 bit: 1092091904 Stream Length: 8 - +

Float 32 bit: 9,500000e+00 Float 64 bit: 5,395651e-315

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x4

Como visualizar um arquivo binário?

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id;
5     float nota;
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita binário (W
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

O TIPO do dado informa como devemos interpretar os BYTES!

Inteiro (4 bytes) = 19412123

dados.bin - GHex

File Edit View Windows Help

00000000 9B 34 28 01 00 00 18 41 .4(. .A

Signed 8 bit:	0	Signed 32 bit:	1092091904	Hexadecimal:	00
Unsigned 8 bit:	0	Unsigned 32 bit:	1092091904	Octal:	000
Signed 16 bit:	0	Signed 64 bit:	1092091904	Binary:	00000000
Unsigned 16 bit:	0	Unsigned 64 bit:	1092091904	Stream Length:	8 - +
Float 32 bit:	9,500000e+00	Float 64 bit:	5,395651e-315		

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

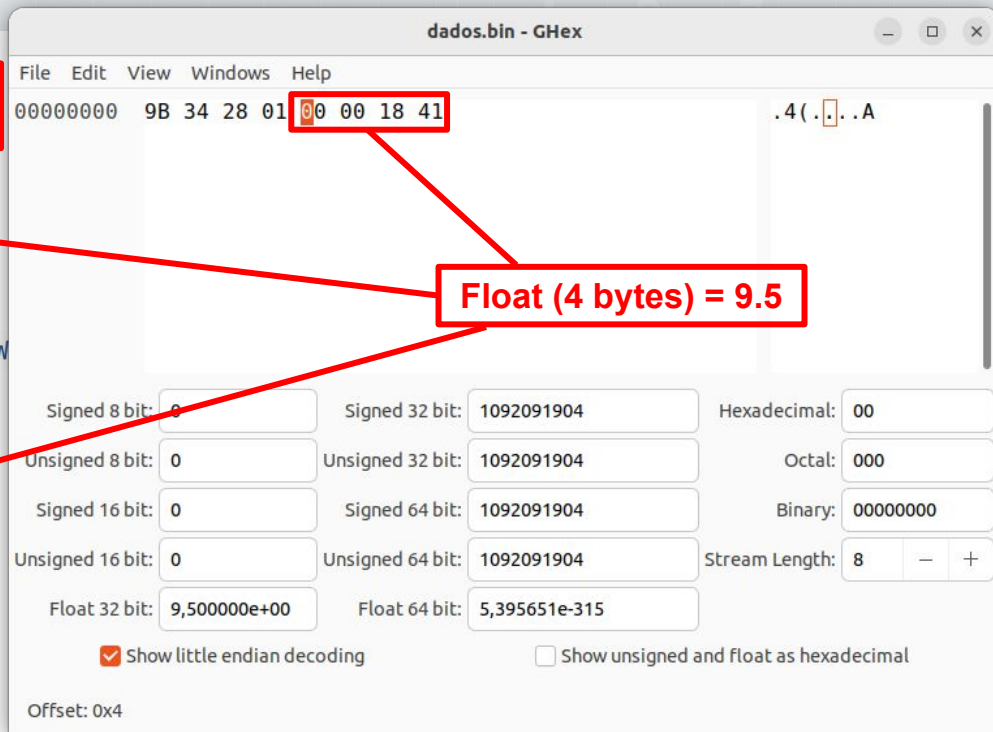
Offset: 0x4

Como visualizar um arquivo binário?

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id;
5     float nota;
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "wb"); // Escrita binário (W
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fwrite(&a1, sizeof(Aluno), 1, fp);
22
23     fclose(fp);
24
25     return 0;
26 }
```

O TIPO do dado informa como devemos interpretar os BYTES!

Float (4 bytes) = 9.5



Lendo Dados com **fread()**

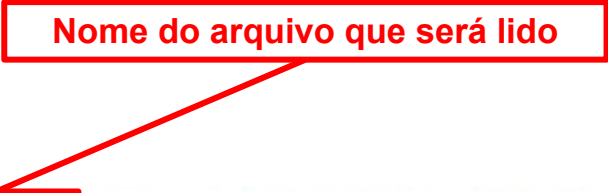
Lendo Dados com **fread()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```

[lendo_arquivo_binario.c](#)

Lendo Dados com **fread()**

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id;
5     float nota;
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a2;
20
21     fread(&a2, sizeof(Aluno), 1, fp);
22
23     printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25     fclose(fp);
26
27     return 0;
28 }
```



[lendo_arquivo_binario.c](#)

Lendo Dados com `fread()`

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```

**Modo de abertura do arquivo
(Leitura Binária - Read Binary)**

[lendo_arquivo_binario.c](#)

Lendo Dados com **fread()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```

Sintaxe muito parecida com o
fwrite()

[lendo_arquivo_binario.c](#)

Lendo Dados com **fread()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```

Endereço de memória que será
copiado os bytes do arquivo

[lendo_arquivo_binario.c](#)

Lendo Dados com **fread()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```

Tamanho do tipo de dado que será lido do arquivo

sizeof(Aluno)

[lendo_arquivo_binario.c](#)

Lendo Dados com **fread()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```

Número de elementos que será lido.

[lendo_arquivo_binario.c](#)

Lendo Dados com **fread()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```

Arquivo que será lido.

[lendo_arquivo_binario.c](#)

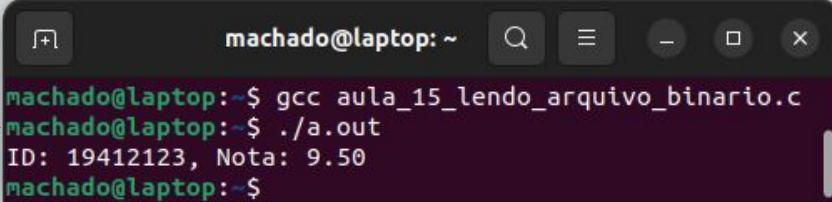
Lendo Dados com **fread()**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a2;
20
21    fread(&a2, sizeof(Aluno), 1, fp);
22
23    printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25    fclose(fp);
26
27    return 0;
28}
```



Lendo Dados com **fread()**

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id;
5     float nota;
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.bin", "rb"); // Leitura binária (Read Binary)
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a2;
20
21     fread(&a2, sizeof(Aluno), 1, fp);
22
23     printf("ID: %d, Nota: %.2f\n", a2.id, a2.nota);
24
25     fclose(fp);
26
27     return 0;
28 }
```



A terminal window titled "machado@laptop: ~" with standard window controls. It shows the compilation and execution of a C program. A red arrow points from the `fread` line in the code block to the terminal output.

```
machado@laptop:~$ gcc aula_15_lendo_arquivo_binario.c
machado@laptop:~$ ./a.out
ID: 19412123, Nota: 9.50
machado@laptop:~$
```

Como seria para escrever o mesmo dados
em arquivo de texto?

Escrevendo em Arquivo: **binário** vs **texto**

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Binary)
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26}
```



Escrevendo em Arquivo: binário vs texto

Binário

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Bina
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26}
```

Texto (ASCII)

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.txt", "w"); // Escrita em texto
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fprintf(fp, "%d %.2f\n", a1.id, a1.nota);
22
23    fclose(fp);
24
25    return 0;
26}
```

Escrevendo em Arquivo: binário vs texto

Binário

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Bina
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26 }
```

Texto (ASCII)

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.txt", "w"); // Escrita em texto
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fprintf(fp, "%d %.2f\n", a1.id, a1.nota);
22
23    fclose(fp);
24
25    return 0;
26 }
```

Escrevendo em Arquivo: binário vs texto

Binário

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.bin", "wb"); // Escrita binário (Write Bina
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fwrite(&a1, sizeof(Aluno), 1, fp);
22
23    fclose(fp);
24
25    return 0;
26 }
```

Texto (ASCII)

```
1#include <stdio.h>
2
3typedef struct {
4    int id;
5    float nota;
6} Aluno;
7
8int main()
9{
10    FILE *fp;
11    fp = fopen("dados.txt", "w"); // Escrita em texto
12
13    if (fp == NULL)
14    {
15        printf("Erro ao abrir o arquivo.\n");
16        return 1;
17    }
18
19    Aluno a1 = {19412123, 9.5};
20
21    fprintf(fp, "%d %.2f\n", a1.id, a1.nota);
22
23    fclose(fp);
24
25    return 0;
26 }
```

Escrevendo em Arquivo: **binário** vs **texto**

**Qual o tamanho do arquivo salvo
em texto (ASCII) ?**

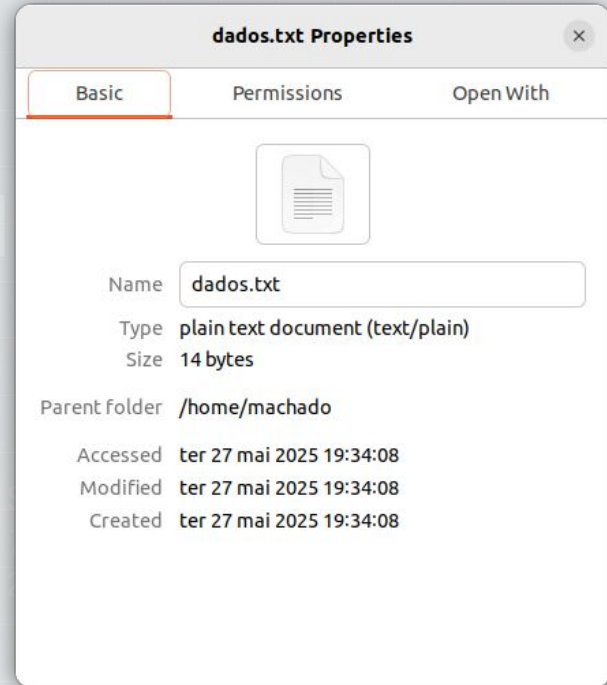
Texto (ASCII)

```
1 #include <stdio.h>
2
3 typedef struct {
4     int id;
5     float nota;
6 } Aluno;
7
8 int main()
9 {
10     FILE *fp;
11     fp = fopen("dados.txt", "w"); // Escrita em texto
12
13     if (fp == NULL)
14     {
15         printf("Erro ao abrir o arquivo.\n");
16         return 1;
17     }
18
19     Aluno a1 = {19412123, 9.5};
20
21     fprintf(fp, "%d %.2f\n", a1.id, a1.nota);
22
23     fclose(fp);
24
25     return 0;
26 }
```


Escrevendo em Arquivo: **binário** vs **texto**

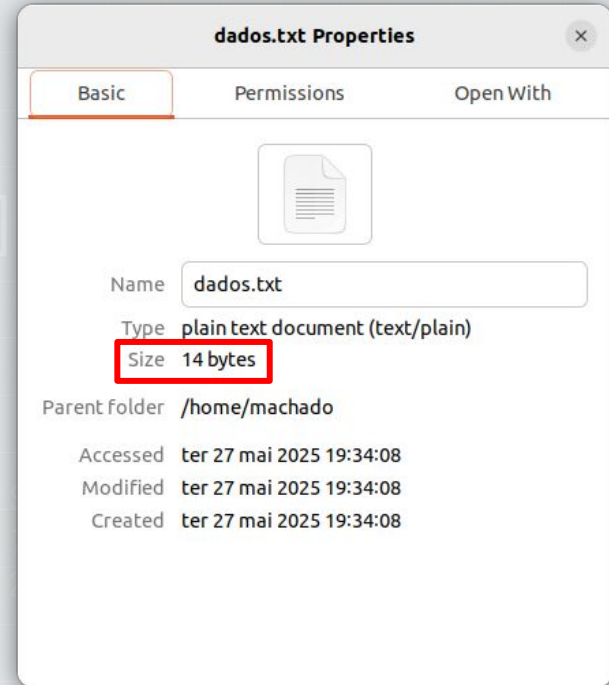
**Qual o tamanho do arquivo salvo
em texto (ASCII) ?**

```
int matrix[3]
```



Escrevendo em Arquivo: **binário** vs **texto**

**Qual o tamanho do arquivo salvo
em texto (ASCII) ?**



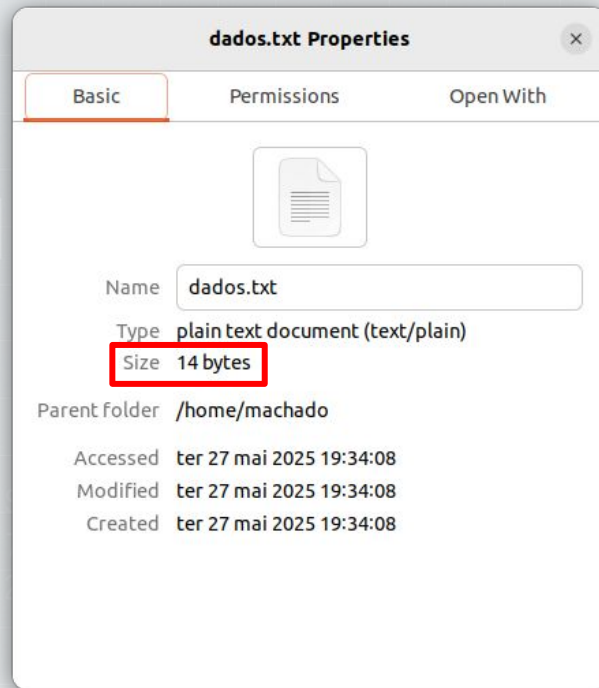
Escrevendo em Arquivo: **binário** vs **texto**

**Qual o tamanho do arquivo salvo
em texto (ASCII) ?**

```
19412123 9.50
```

No Linux: 13 caracteres + `\n` = 14 characters!

No windows: 13 carac. + `\r\n` = 15 bytes!



Escrevendo em Arquivo: **binário** vs **texto**

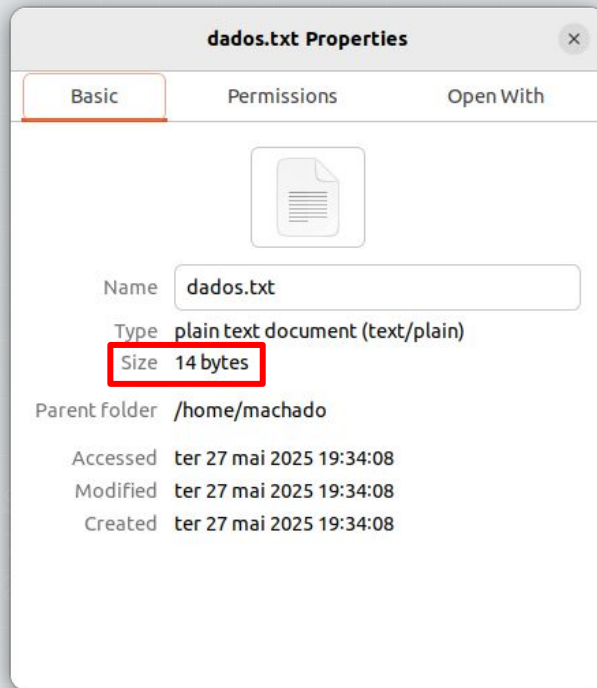
**Qual o tamanho do arquivo salvo
em texto (ASCII) ?**

Não esqueça que espaço também é caracter!

```
19412123 9.50
```

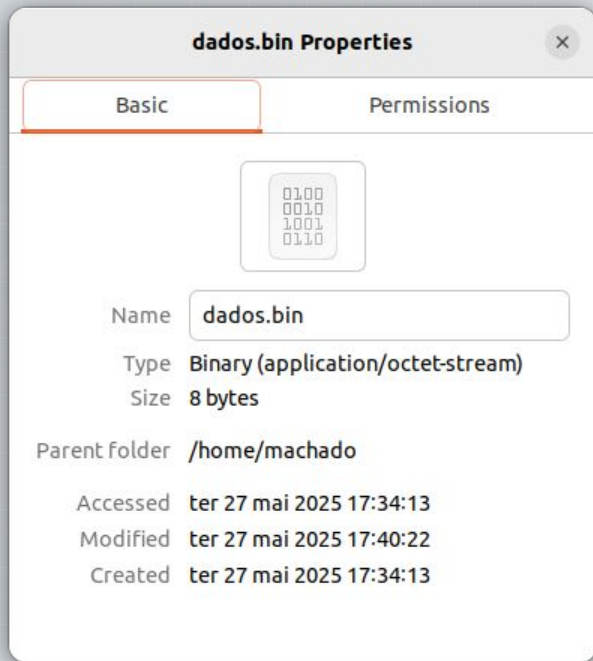
No Linux: 13 caracteres + `\n` = 14 characters!

No windows: 13 carac. + `\r\n` = 15 bytes!

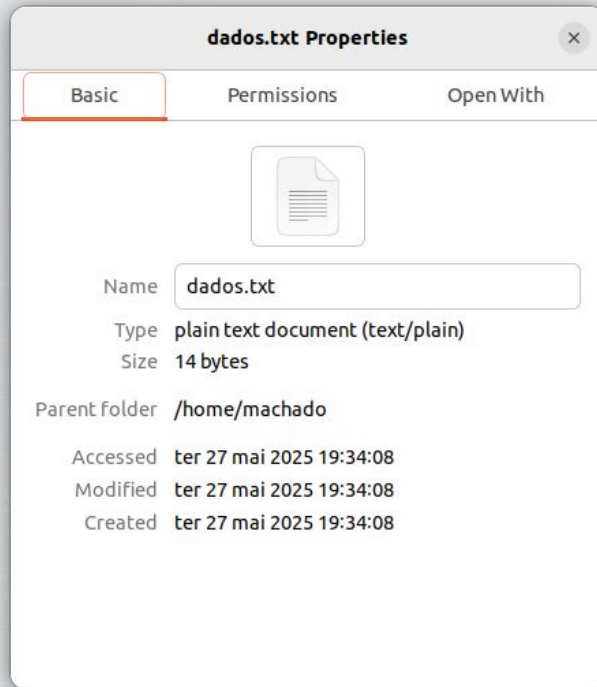


Escrevendo em Arquivo: **binário** vs **texto**

Binário

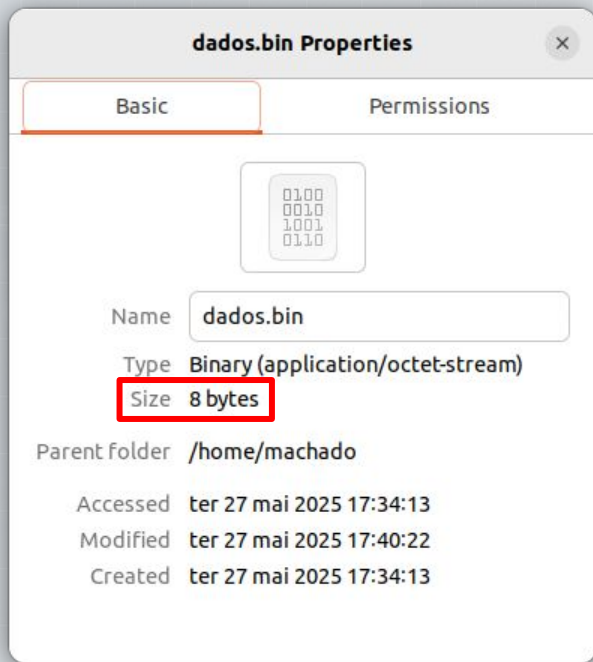


Texto (ASCII)

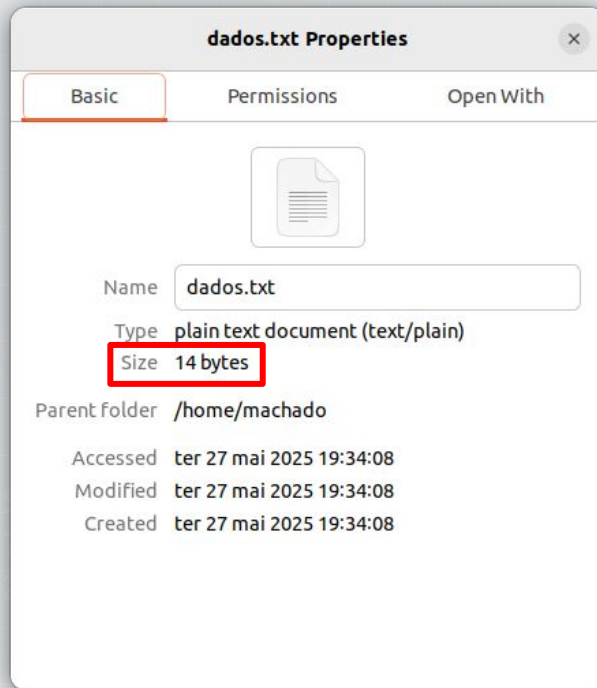


Escrevendo em Arquivo: **binário** vs **texto**

Binário



Texto (ASCII)



Escrevendo em Arquivo: **binário** vs **texto**

Binário

Texto (ASCII)

dados.bin - GHex

File Edit View Windows Help

00000000 9B 34 28 01 00 00 18 41 .4(....A

Signed 8 bit: -101 Signed 32 bit: 19412123 Hexadecimal: 9B

Unsigned 8 bit: 155 Unsigned 32 bit: 19412123 Octal: 233

Signed 16 bit: 13467 Signed 64 bit: 4690499011925783707 Binary: 10011011

Unsigned 16 bit: 13467 Unsigned 64 bit: 4690499011925783707 Stream Length: 8 - +

Float 32 bit: 3,089447e-38 Float 64 bit: 3,932160e+05

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

dados.txt - GHex

File Edit View Windows Help

00000000 31 39 34 31 32 31 32 33 20 39 2E 35 30 0A 19412123 9.50.

Signed 8 bit: 49 Signed 32 bit: 825506097 Hexadecimal: 31

Unsigned 8 bit: 49 Unsigned 32 bit: 825506097 Octal: 061

Signed 16 bit: 14641 Signed 64 bit: 3689065136413489457 Binary: 00110001

Unsigned 16 bit: 14641 Unsigned 64 bit: 3689065136413489457 Stream Length: 8 - +

Float 32 bit: 2,622596e-09 Float 64 bit: 4,422272e-62

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

Escrevendo em Arquivo: **binário** vs **texto**

Binário

Texto (ASCII)

Arquivos Binário são mais **eficientes** em **espaço e tempo de acesso**.

dados.bin - GHex

File Edit View Windows Help

00000000 9B 34 28 01 00 00 18 41 .4(....A

Signed 8 bit: -101 Signed 32 bit: 19412123 Hexadecimal: 9B

Unsigned 8 bit: 155 Unsigned 32 bit: 19412123 Octal: 233

Signed 16 bit: 13467 Signed 64 bit: 4690499011925783707 Binary: 10011011

Unsigned 16 bit: 13467 Unsigned 64 bit: 4690499011925783707 Stream Length: 8 - +

Float 32 bit: 3,089447e-38 Float 64 bit: 3,932160e+05

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

dados.txt - GHex

File Edit View Windows Help

00000000 31 39 34 31 32 31 32 33 20 39 2E 35 30 0A 19412123 9.50.

Signed 8 bit: 49 Signed 32 bit: 825506097 Hexadecimal: 31

Unsigned 8 bit: 49 Unsigned 32 bit: 825506097 Octal: 061

Signed 16 bit: 14641 Signed 64 bit: 3689065136413489457 Binary: 00110001

Unsigned 16 bit: 14641 Unsigned 64 bit: 3689065136413489457 Stream Length: 8 - +

Float 32 bit: 2,622596e-09 Float 64 bit: 4,422272e-62

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

Escrevendo em Arquivo: **binário** vs **texto**

Binário

Texto (ASCII)

Arquivos Binário são mais **difíceis** de manipular e visualizar.

dados.bin - GHex

File Edit View Windows Help

00000000 9B 34 28 01 00 00 18 41 .4(....A

Signed 8 bit: -101 Signed 32 bit: 19412123 Hexadecimal: 9B

Unsigned 8 bit: 155 Unsigned 32 bit: 19412123 Octal: 233

Signed 16 bit: 13467 Signed 64 bit: 4690499011925783707 Binary: 10011011

Unsigned 16 bit: 13467 Unsigned 64 bit: 4690499011925783707 Stream Length: 8 - +

Float 32 bit: 3,089447e-38 Float 64 bit: 3,932160e+05

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

dados.txt - GHex

File Edit View Windows Help

00000000 31 39 34 31 32 31 32 33 20 39 2E 35 30 0A 19412123 9.50.

Signed 8 bit: 49 Signed 32 bit: 825506097 Hexadecimal: 31

Unsigned 8 bit: 49 Unsigned 32 bit: 825506097 Octal: 061

Signed 16 bit: 14641 Signed 64 bit: 3689065136413489457 Binary: 00110001

Unsigned 16 bit: 14641 Unsigned 64 bit: 3689065136413489457 Stream Length: 8 - +

Float 32 bit: 2,622596e-09 Float 64 bit: 4,422272e-62

☒ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0x0

Exercícios!

15.01) Implemente um visualizador de arquivos binários em hexadecimal. A entrada do programa é o nome do arquivo que deve ser aberto e a saída são os bytes do programa em hexadecimal limitados a 16 bytes por linha.

```
int matrix[3][3]:
```

15.02) Escreva um programa que reconheça a assinatura de arquivos binários (ver descrição completa no runcodes).

Prática!

Código Matricula: R4SM

<https://runcodes.icmc.usp.br/offerings/view/83>

[run.codes]



Menu Professor ▾

matheus.m.santos@icmc.usp.br ▾

Hora do Servidor: 14/03/2023 18:46:06

Home > SSC0501

SSC0501 - Introdução à Ciência de Computação I

Professores/Monitores

Professores: Matheus Machado dos Santos
Turma: 2025101
Universidade: USP
Ativa até: 21/07/2025



Código de Matrícula

R4SM



Novo Exercício



Enviar E-mail



Ver Notas



Exportar Tabela de Notas

Exercícios

No.	Exercício	Status	Casos Corretos	Nota	Entregas	Participantes	Prazo de Entrega	Ações
1	Hello World	Finalizado	1/1	10.00	78	43/46	14/03/2025 21:00:00	Ver Detalhes Remover Exercício
2	1.01 Maior número	Não Entregue	0/6	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício
3	1.02 Par ou ímpar	Não Entregue	0/7	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício
4	1.03 Positivo, negativo ou zero	Não Entregue	0/7	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício
5	1.04 Maior de três números	Não Entregue	0/7	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício

SSC0501 - Introdução à Ciência de Computação I

Obrigado pela atenção!!