

SSC0800 - Introdução à Ciência de Computação I

Arquivos

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Baseado no material do prof Cesar Henrique Comin

O que vamos aprender hoje?



Arquivos



Arquivos

- Mecanismo de organização de informação mantida em memória secundária
 - ◆ Acessada via computador
- Disco (HDD)
- Disquete, Fitas Magnéticas
- CD, DVD
- Pen-drive, cartões, SSD
- ...

Arquivos

→ Por que utilizamos arquivos?

- ◆ Armazena grande quantidade de memória a um custo relativamente baixo
- ◆ Armazena memória de maneira não-volátil
- ◆ Permite a persistência dos dados

Arquivos

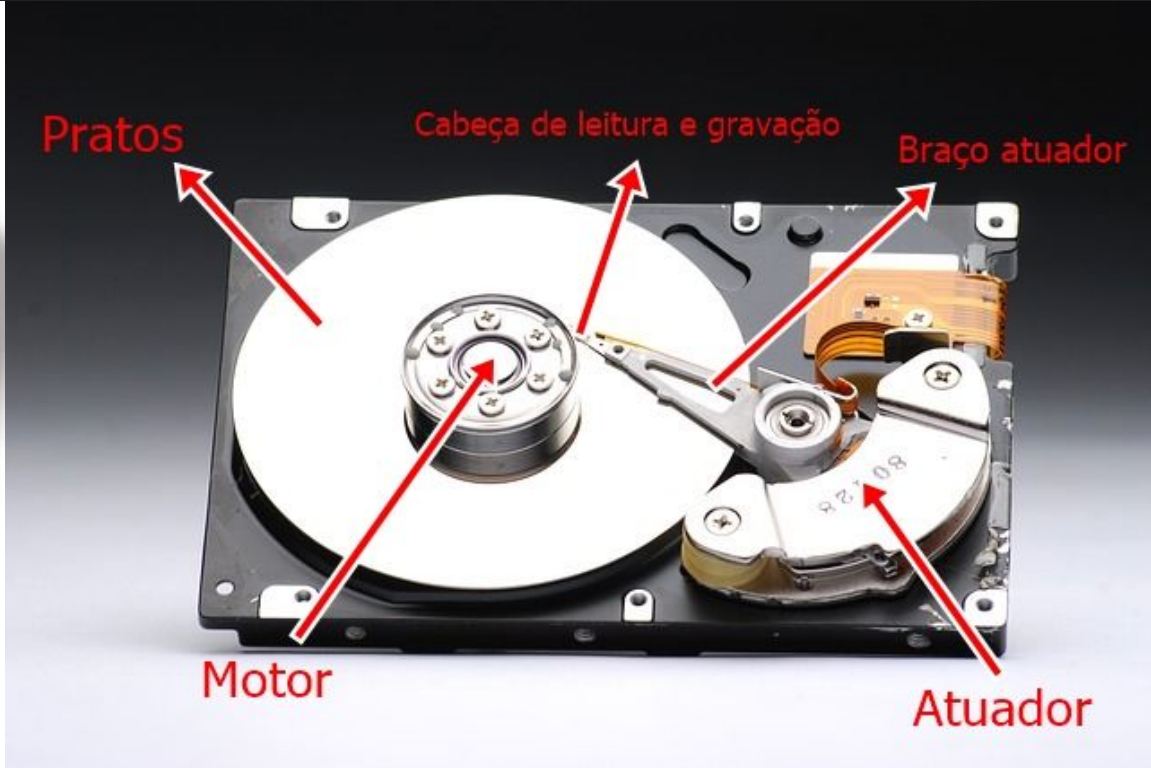
→ Desvantagens

- ◆ Alto tempo de acesso
- ◆ Discos e outros dispositivos de armazenamento secundário são lentos*

*Pode variar bastante

Dispositivos de Armazenamento

0 HDD

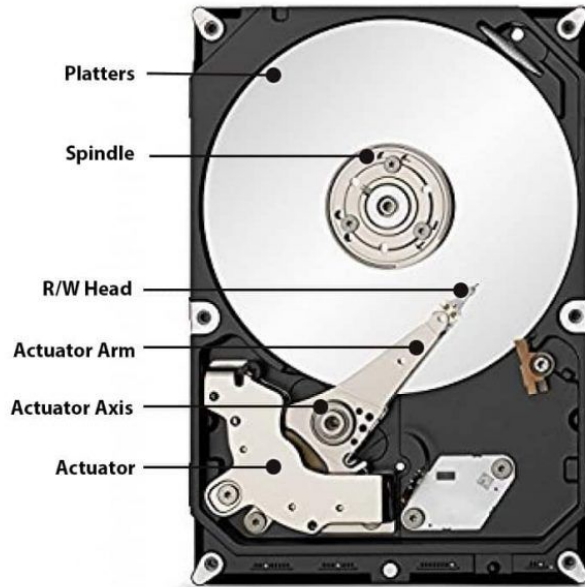


Disco (HDD)

Fonte (Dir.): <https://techenter.com.br/o-que-e-hdd-hard-disk-drive/>

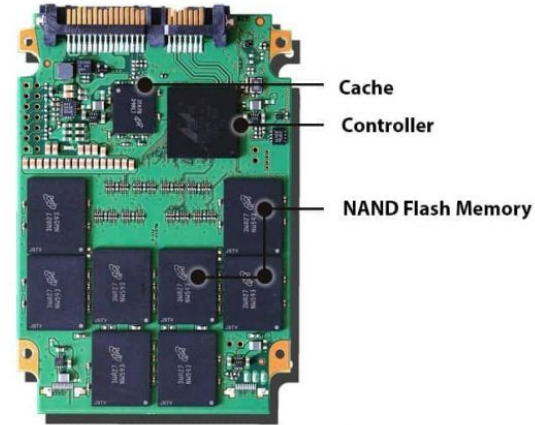
0 SSD

HDD 3.5"



Shock resistant up to 55g (operating)
Shock resistant up to 350g (non-operating)

SSD 2.5"



Shock resistant up to 1500g
(operating and non-operating)

HDD vs SSD (Fonte:

<https://www.leak.pt/ssd-vs-hdd-existe-desenvolvimento-de-hdds/>)



A RAM



Fonte:
<https://i.imgur.com/ALKguuw.jpg>



Fonte:
<https://www.infoescola.com/anfibios/ra-animal/>



Fonte: Chrono
Trigger



RAM

Fontes: <https://br.crucial.com/memory/ddr4/ct8g4dfs824a>

Dispositivos de Armazenamento

→ HDD

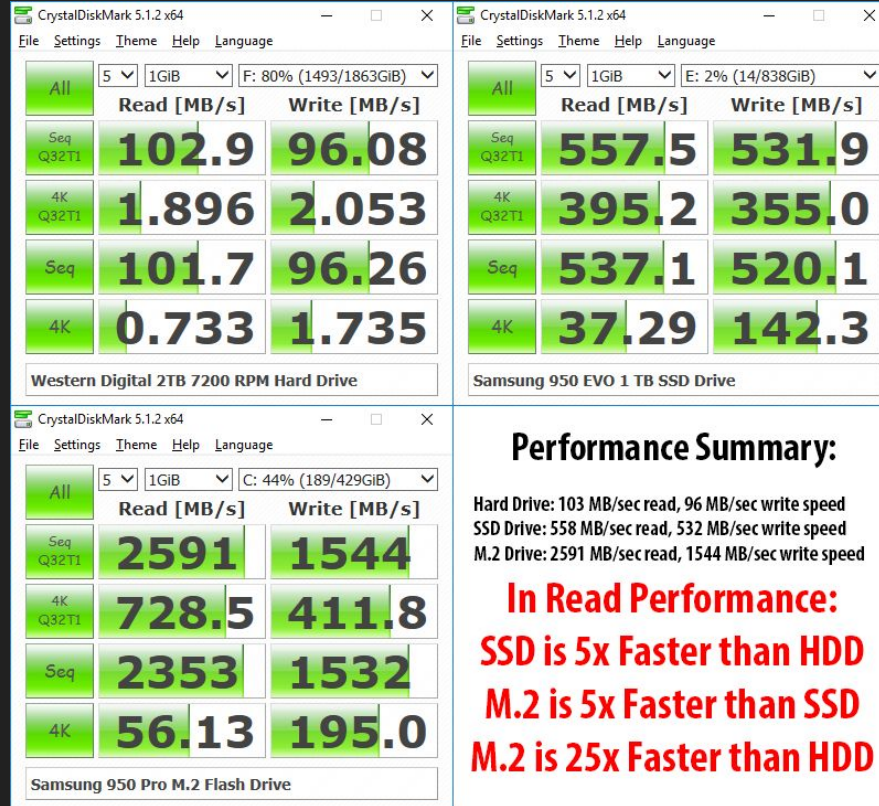
- ◆ Disco mecânico
- ◆ Rotacionar e apontar uma agulha no local adequado

→ SSD

- ◆ Semicondutores, Controlador acessa células NAND
- ◆ Memória NAND ou NVM

→ Ram

- ◆ Circuitos integrados, Semicondutores



Performance Summary:

Hard Drive: 103 MB/sec read, 96 MB/sec write speed
 SSD Drive: 558 MB/sec read, 532 MB/sec write speed
 M.2 Drive: 2591 MB/sec read, 1544 MB/sec write speed

In Read Performance:
SSD is 5x Faster than HDD
M.2 is 5x Faster than SSD
M.2 is 25x Faster than HDD

Fonte: <https://emolike.net/nvme-vs-ssd-vs-hdd-performance>
 Programa usado: <https://crystalmark.info/en/software/crystaldiskmark/>

| Friendly name | Industry name | Peak Transfer Rate | Data transfers/second (in millions) |
|----------------------|----------------------|---------------------------|--|
| DDR4-2400 | PC4-19200 | 19200 MB/s | 2400 |
| DDR4-2666 | PC4-21300 | 21300 MB/s | 2666 |
| DDR4-4000 | PC4-32000 | 32000 MB/s | 4000 |
| DDR4-4400 | PC4-35200 | 35200 MB/s | 4400 |

Velocidade DDR4

Fonte: <https://www.crucial.com/support/memory-speeds-compatibility>

| Friendly name | Industry name | Peak Transfer Rate | Data transfers/second (in millions) |
|----------------------|----------------------|---------------------------|--|
| DDR2-400 | PC2-3200 | 3200 MB/s | 400 |
| DDR2-533 | PC2-4200 | 4266 MB/s | 533 |
| DDR2-667 | PC2-5300 | 5333 MB/s | 667 |
| DDR2-800 | PC2-6400 | 6400 MB/s | 800 |
| DDR2-1000 | PC2-8000 | 8000 MB/s | 1000 |

Velocidade DDR2

Fonte: <https://www.crucial.com/support/memory-speeds-compatibility>

Dispositivos de Armazenamento

- Portanto, mesmo as memórias RAM mais antigas (DDR2) superam os melhores SSDs atuais.
- Para as memórias mais recentes (DDR4), a transferência chega a ser 10x mais rápido!
- E também temos a questão da latência!

Dispositivos de Armazenamento

→ HDD

◆ Alguns milisegundos => $\sim 2\text{ms}-30\text{ms}$ [1, 2];

→ SSD

◆ Alguns microssegundos => $\sim 10\mu\text{s}-100\mu\text{s}$ [3];

→ RAM

◆ Alguns nanossegundos => $\sim 10\text{ns}-30\text{ns}$ [4];

| HDD | SSD | RAM |
|-------------------|-------------------|-------------------|
| 10^{-3}s | 10^{-6}s | 10^{-9}s |

| | RAM | Discos (HDD) | SSD |
|------------------------|--------------|---------------------|---------------|
| Custo | Alto | Baixo | Intermediário |
| Tempo de Acesso | Baixo | Alto | Médio |
| Capacidade | Baixa | Alto | Alta |
| Princípio | Elétrico | Magnético | Elétrico |
| Persistência | Volátil | Não-volátil | Não-volátil |
| Acesso | Aleatório | Aleatório | Aleatório |
| Organização | Células | Trilhas/Setores | Células |

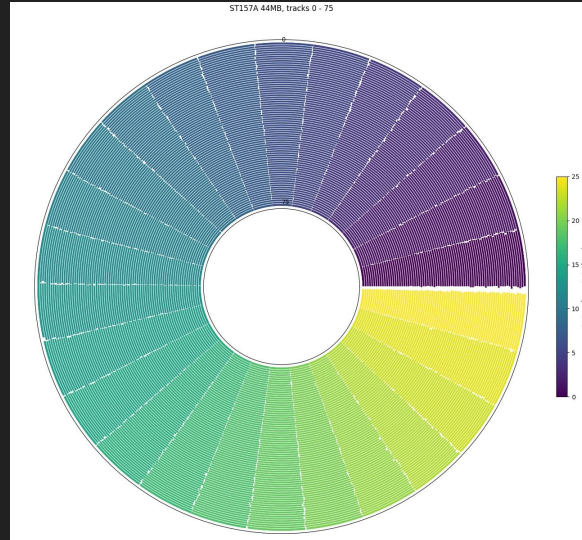
Sistema de Arquivos

Sistema de Arquivos

→ Formatação Física

◆ Organização do disco em setores/trilhas

- Vem da fábrica



Fonte: <https://blog.stuffedcow.net/2019/09/hard-disk-geometry-microbenchmarking/>

Sistema de Arquivos

→ Formatação Lógica

◆ 'Instala' o sistema de arquivos no disco:

- Subdivide o disco em regiões endereçáveis;
- Grava estruturas de gerenciamentos dos arquivos.
- Overhead de espaço ocupado com informações para gerenciamento

Sistema de Arquivos

The screenshot displays the MiniTool Partition Wizard interface. The main window shows a list of partitions with columns for Partition, Capacity, Used, Unused, File System, and Type. A red box highlights the 'Type' column, which contains 'Primary' and 'Logical' options. Below the list, there are visual representations of Disk 1, Disk 2, and Disk 3, showing their respective partitions and capacities.

| Partition | Capacity | Used | Unused | File System | Type |
|--|-----------|-----------|-----------|-------------|------------------|
| *:System Reserved | 549.00 MB | 399.23 MB | 149.77 MB | NTFS | Primary |
| C: | 59.46 GB | 36.26 GB | 23.20 GB | NTFS | Primary |
| D: | 100.00 GB | 107.08 MB | 99.89 GB | NTFS | Primary |
| G: | 100.00 GB | 107.14 MB | 99.89 GB | NTFS | Logical |
| *: | 240.00 GB | 0 B | 240.00 GB | Unallocated | Logical |
| Disk 2 (VMware, VMware Virtual S SAS, GPT, 1.00 TB) | | | | | |
| E: | 512.04 GB | 155.98 MB | 511.89 GB | NTFS | GPT (Data Partit |
| F: | 511.86 GB | 155.97 MB | 511.71 GB | NTFS | GPT (Data Partit |
| *: | 103.98 MB | 0 B | 103.98 MB | Unallocated | GPT |

0 Operations Pending

Disk 1 (MBR, 500.00 GB): System Res (549 MB Use), C:(NTFS) (59.5 GB Use), D:(NTFS) (100.0 GB (Used: C), G:(NTFS) (100.0 GB (Used: C), (Unallocated) 240.0 GB

Disk 2 (GPT, 1.00 TB): E:(NTFS) (512.0 GB (Used: 0%), F:(NTFS) (511.9 GB (Used: 0%), (Unallocated) 104 MB

Disk 3 (MBR, 500.00 GB): (Unallocated) 500.0 GB

Legend: Primary/Logical, Simple, Spanned, Striped, Mirrored, RAID5, Unallocated

Fonte: <https://www.partitionwizard.com/partitionmanager/primary-partition-vs-logical-drive.html>

Sistema de Arquivos

→ Sistema de Arquivos

- ◆ Faz parte do sistema operacional (S.O.)
- ◆ Fornece a infraestrutura básica para a manipulação de arquivos em memória secundária via software
- ◆ Oferece um conjunto de operações para a manipulação de arquivos

Operações Sistema de Arquivos

| | |
|------------------------|----------------------------------|
| criar (create, open) | destruir ou remover (delete) |
| renomear (rename) | abrir (open) |
| fechar (close) | ler dados (read) |
| escrever dados (write) | escrever dados no final (append) |
| posicionar (seek) | ... |



Arquivos Físico e Lógico

Arquivos Físico e Lógico

→ Arquivo Físico:

◆ Sequência de bytes armazenada no disco em blocos distribuídos em trilhas/setores.

→ Arquivo Lógico:

◆ Arquivo como visto pelo aplicativo que o acessa – sequência contínua de registros ou bytes.

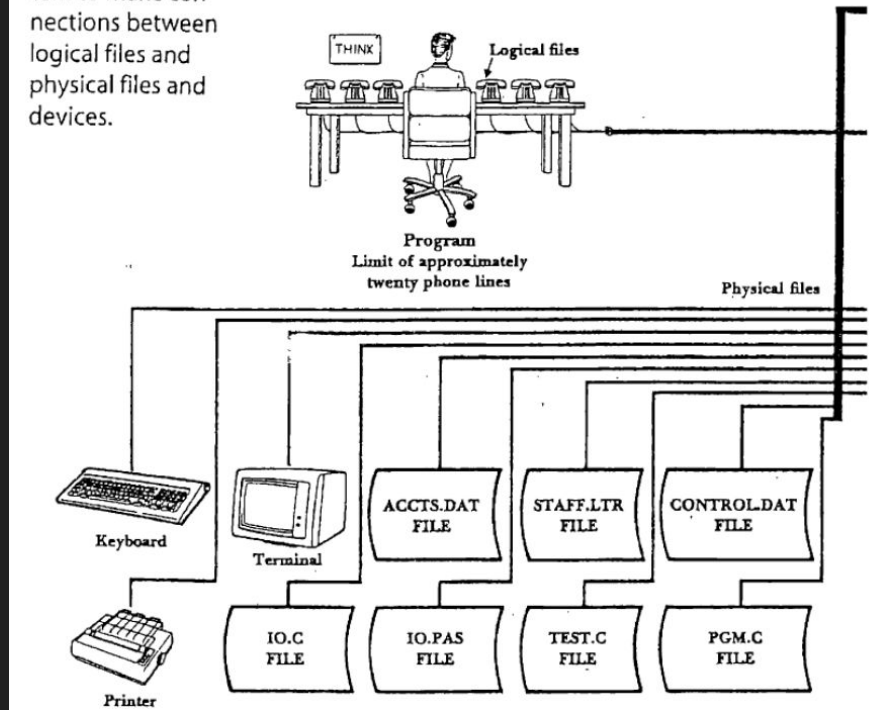
→ Associação arquivo físico/arquivo lógico:

◆ Gerenciada pelo Sistema de Arquivo/S.O.

Arquivos Físico e Lógico

- Para uma aplicação, um arquivo é como uma conexão de telefone em uma rede de telefones
- O programa pode receber e enviar bytes pela linha
 - ◆ Mas não sabe para onde eles vão ou da onde eles vem
- O programa sabe apenas sobre o fim da sua linha
- Essa linha é o “arquivo lógico”

Figure 2.1 The program relies on the operating system to make connections between logical files and physical files and devices.



Analogia de arquivo lógico e físico. Fonte: [1]

Arquivo Físico

→ Sequência de bytes

| | | | | | | | | | | | | | | | |
|----|----|-----|------|------|------|-----|------|------|------|-----|-------|-------|-------|-----|-------|
| 0 | 1 | | 4095 | 4096 | 4097 | | 8191 | 8192 | 8193 | | 12287 | 12288 | 12289 | | 16383 |
| B0 | 80 | ... | D1 | 28 | 19 | ... | 01 | 00 | 13 | ... | 2A | 3E | 1F | ... | 33 |

byte 0 = 1^o byte

byte 1 = 2^o byte



byte n = (n+1) byte

tamanho:
16.384 bytes ou 16KB

Arquivo Lógico

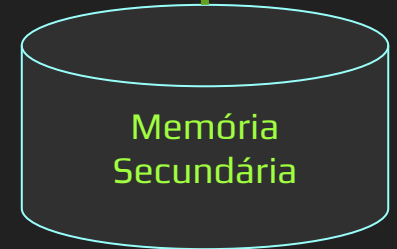
Arquivo Lógico

- Para a maioria das linguagens de programação:
 - ◆ Informações lidas e gravadas em arquivos como streams de bytes.
 - ◆ Gerenciar arquivos é gerenciar esses streams.
- Não importa qual a fonte e organização do arquivo físico!
- Tudo isso é abstraído pelo SO (ainda bem)

Dados de entrada
para
processamento



Ler dados da
memória
secundária



Dados de saída
para salvar
quando buffer for
"descarregado"



Escrever dados na
memória
secundária

Fluxo de dados

Fechamento de Arquivos

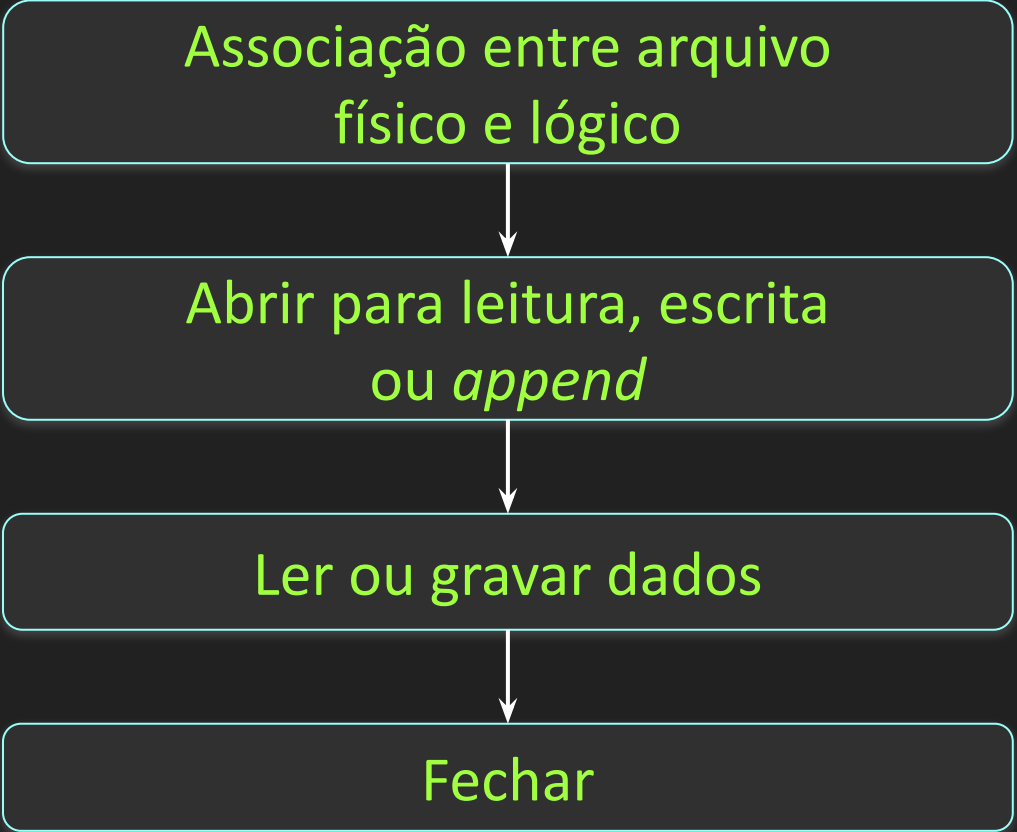
- Encerra a associação entre arquivos lógico e físico
 - ◆ Todas as informações são atualizadas e salvas
 - ◆ Conteúdo dos buffers de E/S enviados para o arquivo.

Fechamento de Arquivos

- S.O. fecha o arquivo se o aplicativo não o fizer ao final da execução do programa.
- Interessante para:
 - ◆ Garantir que os buffers sejam descarregados;
 - ◆ Liberar as estruturas associadas ao arquivo para outros arquivos.
- É importante fechar ainda assim para evitar perdas de dados por interrupção e liberar os nomes lógicos cedo

Fechamento de Arquivos

```
int main ()
{
    FILE * pFile;
    pFile = fopen ("myfile.txt", "w");
    if (pFile != NULL)
    {
        fclose (pFile);
    }
    return 0;
}
```



```
graph TD; A[Associação entre arquivo físico e lógico] --> B[Abrir para leitura, escrita ou append]; B --> C[Ler ou gravar dados]; C --> D[Fechar];
```

Associação entre arquivo físico e lógico

Abrir para leitura, escrita ou *append*

Ler ou gravar dados

Fechar

Operações Sobre Arquivos

Aplicações

Aplicações

- O uso de arquivos na memória secundária são vastos
- Em geral, podem ser resumidos a:
 - ◆ Permanência dos dados
 - Não são deletados ao final do programa
 - ◆ Salvar dados maiores
 - Limite muito maior que o da RAM
 - ◆ Portabilidade dos dados
 - Pode-se transferir dados entre dispositivos

Aplicações

→ Alguns exemplos de uso de arquivos

◆ Salvar dados das aplicações

- Logs, resultados, backups, perfis, histórico, etc.

◆ Salvar arquivos multimídia

- Imagens, vídeos, músicas, etc.

◆ Jogos

- Progresso, modelos 3D, texturas, sons, etc.

/home/tilman/PNG-Gradient.png - Bless

File Edit View Search Tools Help

PNG-Gradient.png ✕

| | | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| 00000000 | 89 | 50 | 4E | 47 | 0D | 0A | 1A | 0A | 00 | 00 | 00 | 0D | 49 | 48 | 44 | 52 | 00 | .PNG.....IHDR. |
| 00000011 | 00 | 00 | 80 | 00 | 00 | 00 | 44 | 08 | 02 | 00 | 00 | 00 | C6 | 25 | AA | 3E | 00 |D.....%.>. |
| 00000022 | 00 | 00 | C2 | 49 | 44 | 41 | 54 | 78 | 5E | ED | D4 | 81 | 06 | C3 | 30 | 14 | 40 | ...IDATx^.....0.@ |
| 00000033 | D1 | B7 | 34 | DD | FF | FF | 6F | B3 | 74 | 56 | EA | 89 | 12 | 6C | 28 | 73 | E2 | ..4...o.tV...l(s. |
| 00000044 | AA | 34 | 49 | 03 | 87 | D6 | FE | D8 | 7B | 89 | BB | 52 | 8D | 3B | 87 | FE | 01 | .4I.....{.R;...; |
| 00000055 | 00 | 80 | 00 | 00 | 10 | 00 | 00 | 02 | 00 | 40 | 00 | 00 | 08 | 00 | 00 | 01 | 00 |@..... |
| 00000066 | 20 | 00 | 00 | 04 | 00 | 80 | 00 | 00 | 10 | 00 | 00 | 02 | 00 | 40 | 00 | 00 | 08 |@..... |
| 00000077 | 00 | 00 | 01 | 00 | 20 | 00 | 00 | 00 | D4 | 5E | 6A | 64 | 4B | 94 | F5 | 98 | 7C |^jdK... |
| 00000088 | D1 | F4 | 92 | 5C | 5C | 3E | CF | 9C | 3F | 73 | 71 | 58 | 5F | AF | 8B | 79 | 5B | ...\\>..?sqX...y[|
| 00000099 | EE | 96 | B6 | 47 | EB | F1 | EA | D1 | CE | B6 | E3 | 75 | 3B | E6 | B9 | 95 | 8D | ...G.....u;...; |
| 000000aa | C7 | CE | 03 | 39 | C9 | AF | C6 | 33 | 93 | 7B | 66 | 37 | CF | AB | BF | F9 | C9 | ...9...3.{f7..... |
| 000000bb | 2F | 08 | 80 | 00 | 00 | 10 | 00 | 00 | 02 | 00 | 40 | 00 | 00 | 08 | 00 | 00 | 01 | /.....@..... |
| 000000cc | 00 | 20 | 00 | 00 | 04 | 00 | 80 | 00 | 00 | 10 | 00 | 00 | 02 | 00 | 40 | 00 | 00 |@..... |
| 000000dd | 08 | 00 | 00 | 01 | 00 | 20 | 00 | 00 | 8C | 37 | DB | 68 | 03 | 20 | FB | ED | 96 |7.h. ... |
| 000000ee | 65 | 00 | 00 | 00 | 00 | 49 | 45 | 4E | 44 | AE | 42 | 60 | 82 | | | | | e....IEND.B`. |

Signed 8 bit: Signed 32 bit: Hexadecimal: ✕
 Unsigned 8 bit: Unsigned 32 bit: Decimal:
 Signed 16 bit: Float 32 bit: Octal:
 Unsigned 16 bit: Float 64 bit: Binary:
 Show little endian decoding Show unsigned as hexadecimal ASCII Text:
 Offset: 0x4 / 0xfa Selection: 0x1 to 0x3 (0x3 bytes) INS

Uma imagem PNG. Fonte: <https://en.wikipedia.org/wiki/PNG>



Uma imagem PNG. Fonte: <https://en.wikipedia.org/wiki/PNG>



Arquivo de progresso do Stardew Valley. Fonte: <https://stardewvalleywiki.com/Saves>

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <SaveGame xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3    <player>
4      <name>Jesse</name>
5      <isEmoting>>false</isEmoting>
6      <isCharging>>false</isCharging>
7      <willDestroyObjectsUnderfoot>>true</willDestroyObjectsUnderfoot>
8      <isGlowing>>false</isGlowing>
9      <coloredBorder>>false</coloredBorder>
10     <flip>>false</flip>
11     <drawOnTop>>false</drawOnTop>
12     <faceTowardFarmer>>false</faceTowardFarmer>
13     <faceAwayFromFarmer>>false</faceAwayFromFarmer>
14     <ignoreMovementAnimation>>false</ignoreMovementAnimation>
15     <scale>1</scale>
16     <timeBeforeAIMovementAgain>0</timeBeforeAIMovementAgain>
17     <glowingTransparency>0.7000001</glowingTransparency>
18     <glowRate>0.05</glowRate>
19     <Position>
20       <X>1760</X>
21       <Y>864</Y>
22     </Position>
23     <Speed>5</Speed>
24     <IsEmoting>>false</IsEmoting>
25     <CurrentEmote>24</CurrentEmote>
26     <questLog>
27       <Quest>
28         <_currentObjective>&quot;Give the sand dragon his final meal.&quot;</_currentObjective>
29         <_questDescription>You've found yet another strange note within the Mayor's fridge. This time, the instructions are more cryptic.</_questDescription>
30         <_questTitle>Der mysteriöse Qi</_questTitle>
31         <rewardDescription>-1</rewardDescription>
32         <accepted>>false</accepted>
33         <completed>>false</completed>
34         <dailyQuest>>false</dailyQuest>
35         <showNew>>false</showNew>
36         <canBeCancelled>>false</canBeCancelled>
37         <destroy>>false</destroy>
38         <id>4</id>
39         <moneyReward>0</moneyReward>
40         <questType>1</questType>
41         <daysLeft>0</daysLeft>
42         <nextQuests>
43           <int>-1</int>
44         </nextQuests>
45         <questTitle>The Mysterious Qi</questTitle>
46       </Quest>

```

Arquivo de progresso do Stardew Valley. Fonte: <https://stardewvalleywiki.com/Saves>



Vamos Praticar

Base de Dados

→ [https://www.kaggle.com/datasets/hamzacyberpatcher/
data-of-1010-pokemons/](https://www.kaggle.com/datasets/hamzacyberpatcher/data-of-1010-pokemons/)




Colab



Arquivos em Python

Arquivos em Python

- Suponha que temos em um diretório do computador o arquivo “texto.txt”, contendo informações que queremos processar utilizando um programa Python.
- Como podemos ler o conteúdo do arquivo em um programa?

| Nome | Data de modificação | Tipo | Tamanho |
|---|---------------------|--------------------|---------|
|  le_arquivo.py | 19/11/2017 17:52 | Arquivo PY | 1 KB |
|  texto.txt | 19/11/2017 17:47 | Documento de Te... | 1 KB |

Arquivos em Python

- Primeiro passo: criar um programa Python na mesma pasta onde o arquivo se encontra
- Segundo passo: utilizar a função open

arquivo é uma variável do tipo *file*

Carrega o arquivo **texto.txt** para leitura (o caractere '**r**' significa *read*)

```
arquivo = open('texto.txt', 'r')
texto = arquivo.read()
arquivo.close()

print(texto)
```

Lê todo o conteúdo do arquivo, e armazena na variável **texto**

Fecha o arquivo

Arquivos em Python

→ É muito importante lembrar de fechar o arquivo ao terminar de utilizá-lo! Se o arquivo não for fechado, ele pode ficar “preso” pelo sistema operacional.

Arquivos em Python

- No Windows, dependendo de como ele foi configurado, pode ser que a extensão do arquivo esteja escondida.
- Se isso ocorrer, o arquivo texto.txt aparecerá na pasta apenas como texto
 - ◆ Nesse caso, ainda precisamos utilizar o nome de arquivo juntamente com a extensão dele para realizar a leitura

Arquivos em Python

→ Caso a extensão esteja escondida, é um erro comum o arquivo ser nomeado como `texto.txt.txt`

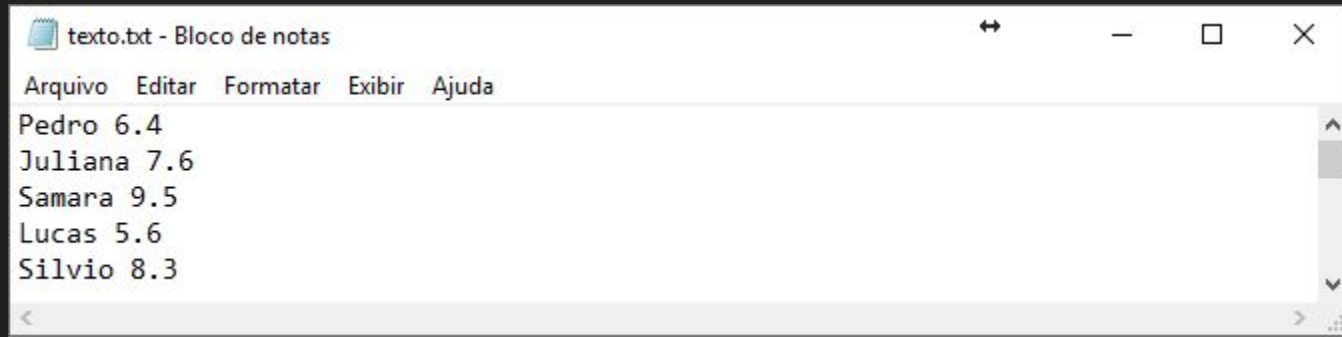
Arquivo de texto com a extensão sendo mostrada

| Nome | Data de modificaç... | Tipo | Tamanho |
|---|----------------------|--------------------|---------|
|  texto.txt | 25/10/2018 19:58 | Documento de Te... | 0 KB |

Arquivo de texto com a extensão escondida

| Nome | Data de modificaç... | Tipo | Tamanho |
|---|----------------------|--------------------|---------|
|  texto | 25/10/2018 19:58 | Documento de Te... | 0 KB |

Nesse caso, se nomeássemos o arquivo como **texto.txt**, ele na verdade seria salvo como **texto.txt.txt**



```
arquivo = open('texto.txt', 'r')  
texto = arquivo.read()  
arquivo.close()  
  
print(texto)
```

```
Pedro 6.4  
Juliana 7.6  
Samara 9.5  
Lucas 5.6  
Silvio 8.3
```

Fonte: Slides do Prof. Cesar Henrique Comin

```
>>> texto
'Pedro 6.4\nJuliana 7.6\nSamara 9.5\nLucas 5.6\nSilvio 8.3\n'
>>> print(texto)
Pedro 6.4
Juliana 7.6
Samara 9.5
Lucas 5.6
Silvio 8.3
```

Fonte: Slides do Prof. Cesar Henrique Comin

Arquivos em Python

- A variável texto na realidade possui a string indicada no primeiro comando.
- Ao utilizarmos a função print , a string contida em texto é modificada para ser mostrada na tela de forma mais intuitiva.
- O caractere '\n' é um caractere especial, chamado de caractere de controle. Ele marca o fim de uma linha.

Arquivos em Python

- E se quisermos separar as linhas do arquivo?
- Abordagem 1, utilizando o método `splitlines`, que opera sobre strings

```
arquivo = open('texto.txt', 'r')
texto = arquivo.read()
arquivo.close()
linhas = texto.splitlines()

print(linhas)
```

```
['Pedro 7.8', 'Juliana 4.5', 'Samara 9.4', 'Lucas 7.4', 'Silvio 6.4']
```

Arquivos em Python

- E se quisermos separar as linhas do arquivo?
- Abordagem 2, utilizando o método `readlines`, que opera sobre arquivos:

```
arquivo = open('texto.txt', 'r')
texto = arquivo.readlines()
arquivo.close()

print(texto)
```

```
['Pedro 6.4\n', 'Juliana 7.6\n', 'Samara 9.5\n', 'Lucas 5.6\n', 'Silvio 8.3\n']
```

Arquivos em Python

- E se quisermos separar as linhas do arquivo?
- Abordagem 3, utilizando estrutura de repetição:

```
arquivo = open('texto.txt', 'r')
texto = []
for linha in arquivo:
    texto.append(linha)
arquivo.close()

print(texto)
```

```
['Pedro 6.4\n', 'Juliana 7.6\n', 'Samara 9.5\n', 'Lucas 5.6\n', 'Silvio 8.3\n']
```

Escrita de arquivo

- Utilizamos novamente a função open, mas agora passamos o parâmetro 'w' ao invés de 'r'.
- Também utilizamos a função write
- Se o arquivo não existir, ele é automaticamente criado. Se ele existir, o conteúdo do arquivo é apagado


```
texto = "Esse texto sera escrito no arquivo"  
  
arquivo = open('mensagem.txt', 'w')  
arquivo.write(texto)  
arquivo.close()
```

O caractere 'w'
significa *write*

Escrita de arquivo

- E se quisermos escrever diversas linhas?
- Utilizamos o caractere '\n'

```
texto = "Pedro\nSamara\nAline\nClara\nSilvio\n"  
  
arquivo = open('mensagem.txt', 'w')  
arquivo.write(texto)  
arquivo.close()
```

Escrita de arquivo

- E se quisermos escrever diversas linhas?
- Se tivermos uma lista de strings , precisamos utilizar uma estrutura de repetição para escrever cada linha:

```
nomes = ["Pedro", "Samara", "Aline", "Clara", "Silvio"]

arquivo = open('mensagem.txt', 'w')
for i in range(len(nomes)):
    nome = nomes[i]
    arquivo.write(nome+'\n')
arquivo.close()
```

Escrita de arquivo

- Vimos que ao abrirmos o arquivo para escrita, o texto contido no arquivo é apagado. Para evitarmos isso, podemos abrir o arquivo em modo append , utilizando o parâmetro 'a'
- Nesse caso, o conteúdo que for escrito no arquivo é adicionado ao final do mesmo.

```
arquivo = open("mensagens.txt", "a")
```

Referências

1. M. J. Folk, B. Zoellick and G. Riccardi. File Structures: An object-oriented approach with C++, Addison Wesley, 1998.
2. <https://www.learnpython.org/>
3. <https://www.w3schools.com/python/>
4. <https://panda.ime.usp.br/cc110/static/cc110/index.html>
5. https://www.youtube.com/playlist?list=PLcoJJSvnDgcKpOi_UeneTNTIVOigRQwcn