Persistência de Arquivos: texto, binário, CSV, propriedades

QXD0099 - Desenvolvimento de Software para Persistência

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br

Slides baseados no material do Professor Régis Magalhães

Agenda

- Tipos de Arquivos
- Fluxos
- Leitura de arquivo
- Traduzindo de determinada codificação para unicode
- BOM - Byte Order Mark
- Código Python para lidar com BOM
- Detecção manual da BOM (sem utf-8-sig)
- Lendo uma linha inteira
- Lendo o arquivo inteiro
- Lendo Strings do Teclado
- Escrita em arquivo
- Escrita em arquivo utilizando print

Agenda

- Lendo Strings do Teclado e salvando em um arquivo
- CSV - Comma-separated values
- TSV - Tab-separated values
- Planilhas
- Arquivos de propriedades
- Manipulação de grandes arquivos
- Checksum / Hash de grandes arquivos
- Armazenando vários arquivos em um só arquivo (arquivamento)
- Compressão de arquivos
- Encriptação de arquivos
- Leitura de arquivo ZIP

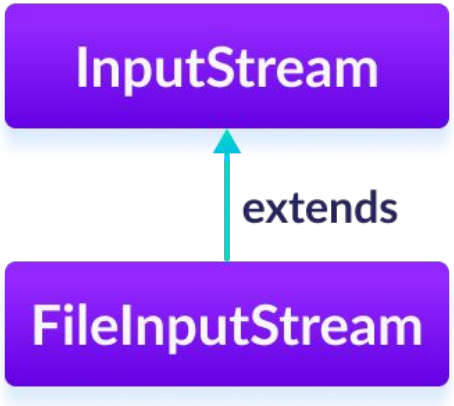
Tipos de Arquivos

- Texto
  - Texto plano
  - Propriedades
  - CSV
  - XML
  - JSON
  - Código fonte

- Binário
  - Imagem
  - Vídeo
  - Áudio
  - Arquivo compactado
  - Código compilado: Executável / Bytecode.
  - PDF

Fluxos

- **Fluxo de Entrada**
  - [Python] - file-like objects ou stream objects - Ler bytes.
  - [Java] - InputStream
- **Fluxo de Saída**
  - [Python] - file-like objects ou stream objects – Escrever bytes.
  - [Java] - OutputStream
- **Servem para operações sobre:**
  - Arquivos.
  - Conexão remota via Socket.
  - Entrada e saída padrão (teclado e console).



Leitura de arquivo

```
# Exemplo de leitura de arquivo em Python

# Abre o arquivo "arquivo.txt" para leitura em modo binário
with open('arquivo.txt', 'rb') as file: # 'rb' significa leitura em modo binário
    b = file.read(1) # Lê o primeiro byte do arquivo

# Exibe o valor do byte lido
print(b)
```

Traduzindo de determinada codificação para unicode

```
# Exemplo de leitura de um arquivo com codificação específica e conversão para Unicode

# Abre o arquivo "arquivo.txt" para leitura com a codificação UTF-8 (ou outra se necessário)
with open('arquivo.txt', 'r', encoding='utf-8') as file:  # Aqui 'utf-8' pode ser substituído por 'iso-8859-1' se necessário
    c = file.read(1)  # Lê o primeiro caractere do arquivo

# Exibe o caractere lido
print(c)
```

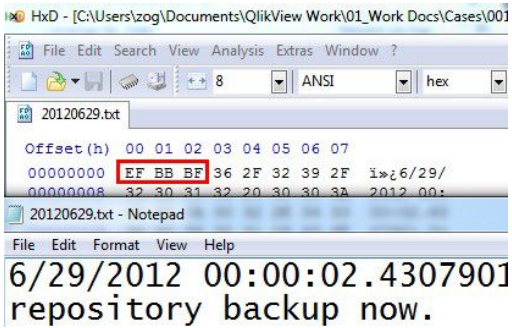
- **`open('arquivo.txt', 'r', encoding='utf-8')`**: Em Python, ao abrir arquivos de texto, você pode especificar a codificação com o argumento **`encoding`**. Aqui, usamos **`'utf-8'`**, mas pode ser substituído por **`'iso-8859-1'`** ou qualquer outra codificação desejada.
- **`file.read(1)`**: Lê o primeiro caractere do arquivo. Ao usar uma codificação, os dados são automaticamente convertidos para Unicode internamente em Python.
- Python já trata a manipulação de **`char`** (caracteres) de maneira nativa, diferente do Java, que precisa de classes específicas como **`InputStreamReader`**.

BOM - Byte Order Mark

- É um uso particular do caractere Unicode especial, U FEFF ZERO WIDTH NO-BREAK SPACE, cuja aparência como um número mágico no início de um fluxo de texto pode sinalizar várias coisas para um programa que lê o texto:
  - A ordem de bytes, ou endianness, do fluxo de texto nos casos de codificações de 16 e 32 bits;
  - O fato de a codificação do fluxo de texto ser Unicode, com alto nível de confiança;
  - Qual codificação de caracteres Unicode é usada.

- A representação UTF-8 da BOM é a sequência de bytes (hexadecimal) EF BB BF.

- O Padrão Unicode permite a BOM em UTF-8, mas não exige nem recomenda seu uso.



Código Python para lidar com BOM

```
# Abrindo o arquivo que pode conter uma BOM, utilizando 'utf-8-sig' para lidar com a BOM
automaticamente
with open('arquivo.txt', 'r', encoding='utf-8-sig') as file:
    conteudo = file.read()

# Exibe o conteúdo do arquivo, sem a BOM
print(conteudo)
```

- **`encoding='utf-8-sig'`**: Ao abrir um arquivo com essa codificação, o Python remove automaticamente o BOM se ele estiver presente. Isso é útil ao trabalhar com arquivos UTF-8 que contêm a BOM.
- **`file.read()`**: Lê o conteúdo do arquivo como texto, ignorando a BOM.

Detecção manual da BOM (sem utf-8-sig)

```python
# Abrindo o arquivo no modo binário para verificar os primeiros bytes manualmente
with open('arquivoBom.txt', 'rb') as file:
    primeiro_bytes = file.read(3)  # Lê os primeiros 3 bytes

    # Verifica se a BOM está presente
    if primeiro_bytes == b'\xef\xbb\xbf':
        print("BOM detectada no arquivo!")
        # Ler o restante do arquivo, agora sem a BOM
        conteudo = file.read().decode('utf-8')
    else:
        # Se não houver BOM, volta ao início e lê o arquivo inteiro
        file.seek(0)
        conteudo = file.read().decode('utf-8')

# Exibe o conteúdo do arquivo
print(conteudo)
```

- O código verifica se o arquivo contém uma BOM UTF-8 no início.
- Se a BOM for encontrada, ela é ignorada, e o restante do arquivo é lido e decodificado.
- Se a BOM não estiver presente, o arquivo é lido desde o início.
- O conteúdo do arquivo é então exibido, sem a BOM.

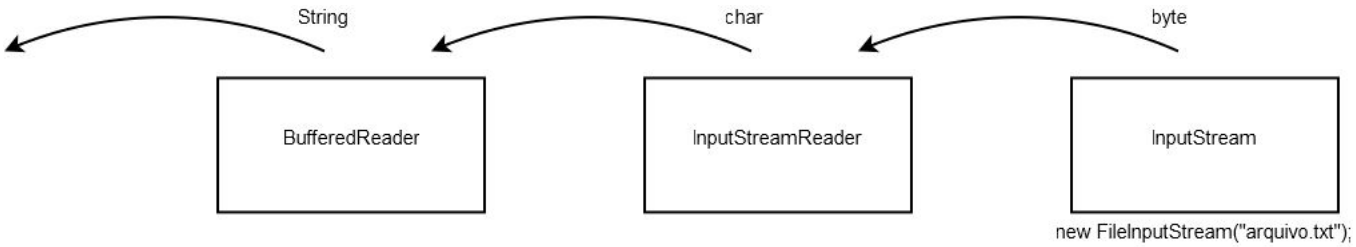
Lendo uma linha inteira

```
# Abrindo o arquivo para leitura de uma linha
with open('arquivo.txt', 'r', encoding='utf-8') as file:
    s = file.readline()   # Lê uma linha do arquivo

# Exibe a linha lida
print(s)
```

- **`open('arquivo.txt', 'r', encoding='utf-8')`**: Abre o arquivo `arquivo.txt` em modo de leitura (`'r'`). O parâmetro `encoding='utf-8'` garante que o arquivo seja lido corretamente como texto UTF-8.
- **`file.readline()`**: Lê uma linha inteira do arquivo. A função `readline()` lê até encontrar um caractere de nova linha (`\n`) ou até o final do arquivo.
- **`print(s)`**: Exibe o conteúdo da linha lida.

Lendo uma linha inteira

Leitura do Reader por pedaços, usando o Buffer, para evitar muitas chamadas ao SO.



Padrão de composição chamado Decorator Pattern.

Lendo o arquivo inteiro

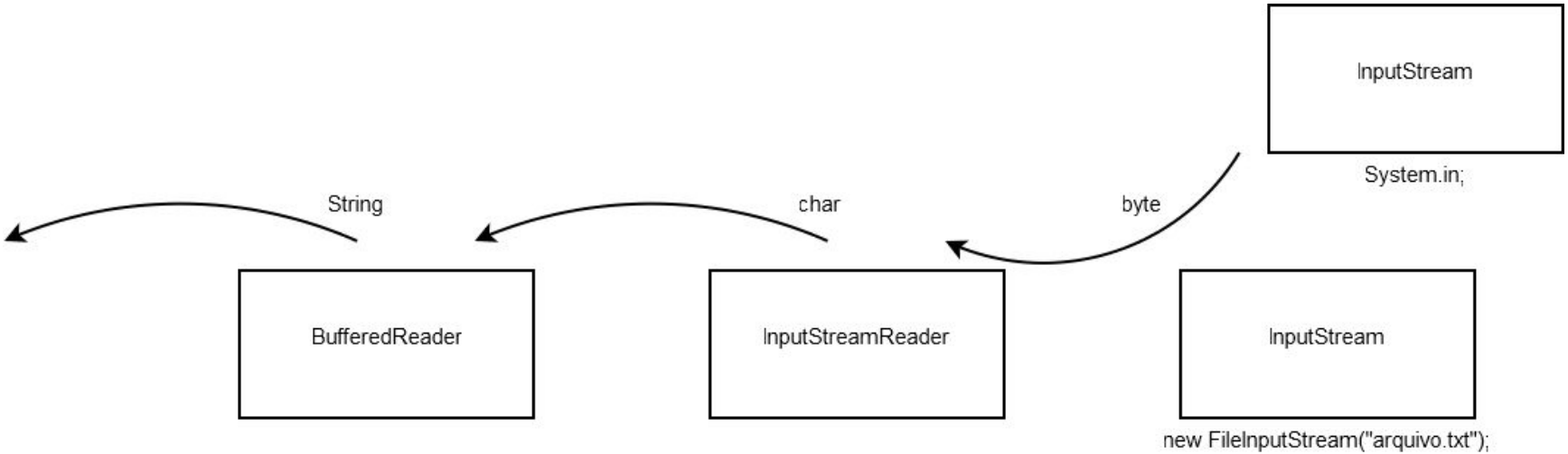
```
# Abrindo o arquivo para leitura
with open('arquivo.txt', 'r', encoding='utf-8') as file:
    # Lê a primeira linha
    s = file.readline()

    # Enquanto houver linhas para ler
    while s:
        # Imprime a linha atual
        print(s.strip())  # .strip() remove os espaços em branco no final da linha, incluindo o \n
        # Lê a próxima linha
        s = file.readline()
```

Lendo Strings do Teclado

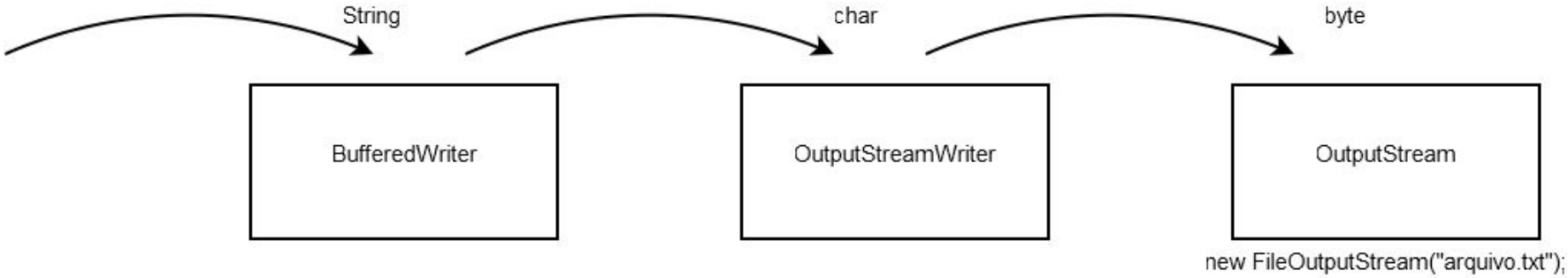
```python
# Leitura da entrada do usuário (console)
import sys

# Lê a primeira linha da entrada padrão
s = sys.stdin.readline().strip()   # .strip() remove espaços em branco e nova linha

# Enquanto houver algo na entrada
while s:
    # Exibe a linha lida
    print(s)
    # Lê a próxima linha da entrada padrão
    s = sys.stdin.readline().strip()
```

- **`sys.stdin.readline()`**: Equivalente ao `br.readLine()` no Java, lê uma linha da entrada padrão (no caso, o console). O `.strip()` remove qualquer nova linha ou espaços em branco no final.
- **`while s:`**: O loop continua até que não haja mais nada a ser lido da entrada, similar ao `while (s != null)` em Java.
- **`print(s)`**: Exibe a linha lida no console.

Lendo Strings do Teclado



Escrita em arquivo



Escrita em arquivo

```
# Abrindo o arquivo para escrita
with open('saida.txt', 'w', encoding='utf-8') as file:
    # Escreve "Java" no arquivo
    file.write("Java\n")
```

- **`open('saida.txt', 'w', encoding='utf-8')`**: Abre o arquivo `saida.txt` para escrita (`'w'`), criando o arquivo se ele não existir. O argumento `encoding='utf-8'` garante que o arquivo seja escrito usando a codificação UTF-8.
- **`file.write("Java\n")`**: Escreve a string "Java" no arquivo, seguida por uma nova linha (`\n`), equivalente ao `bw.newLine()` em Java.
- **`with`**: O uso de `with` garante que o arquivo seja fechado automaticamente após a escrita, eliminando a necessidade de `bw.close()`.

Escrita em arquivo utilizando print

```
# Abre o arquivo "saida.txt" para escrita
with open('saida.txt', 'w', encoding='utf-8') as file:
    # Usa a função print para escrever "Java" no arquivo com uma nova linha automaticamente
    print("Java", file=file)
```

- **`io.StringIO()`**: Cria um buffer em memória que simula um arquivo de texto. Ele age como um stream onde você pode "imprimir" dados.
- **`print("Java", file=buffer)`**: Escreve a string "Java" no buffer, da mesma forma que faria em um arquivo ou com `PrintStream` no Java.
- **`file.write(buffer.getvalue())`**: Escreve o conteúdo do buffer no arquivo real.
- **`buffer.close()`**: Fecha o buffer, embora não seja estritamente necessário para `StringIO`.

Lendo Strings do Teclado e salvando em um arquivo

```
# Abrindo o arquivo para escrita
with open('arquivo.txt', 'w', encoding='utf-8') as file:
    # Lendo strings do teclado
    while True:
        try:
            # Lê uma linha do teclado
            line = input()
            # Escreve a linha no arquivo
            print(line, file=file)
        except EOFError:
            # Termina o loop quando não houver mais
entrada (Ctrl+D no Linux/macOS ou Ctrl+Z no Windows)
            break
```

- **`with open('arquivo.txt', 'w', encoding='utf-8')`**: Abre o arquivo `arquivo.txt` em modo de escrita, usando a codificação UTF-8.
- **`input()`**: Lê uma linha de entrada do teclado.
- **`print(line, file=file)`**: Escreve a linha no arquivo. Isso é equivalente ao `PrintStream.println()` do Java.
- **`EOFError`**: Interrompe o loop quando o usuário envia um sinal de fim de entrada (Ctrl+D em Linux/macOS ou Ctrl+Z no Windows).

CSV - Comma-separated values

- O formato CSV é bastante simples e suportado por quase todas as planilhas eletrônicas e SGDB disponíveis no mercado.
- Cada linha do arquivo representa um registro, e os valores dentro desse registro são separados por vírgulas (ou outro delimitador, como ponto e vírgula, em algumas regiões).
- O formato é amplamente utilizado devido à sua simplicidade e compatibilidade com diversos sistemas e programas, como planilhas e bancos de dados.

CSV - Comma-separated values

Year	Make	Model	Description	Price
1997	Ford	E350	ac, abs, moon	3000.00
1999	Chevy	Venture "Extended Edition"		4900.00
1999	Chevy	Venture "Extended Edition, Very Large"		5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

TSV - Tab-separated values

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	I. setosa
4.9	3.0	1.4	0.2	I. setosa
4.7	3.2	1.3	0.2	I. setosa
4.6	3.1	1.5	0.2	I. setosa
5.0	3.6	1.4	0.2	I. setosa

```
Sepal length Sepal width Petal length Petal width Species
5.1 3.5 1.4 0.2 I. setosa
4.9 3.0 1.4 0.2 I. setosa
4.7 3.2 1.3 0.2 I. setosa
4.6 3.1 1.5 0.2 I. setosa
5.0 3.6 1.4 0.2 I. setosa
```

Planilhas

- Excel
- Google Planilha

- Desenvolvimento de Software Low-Code / No-Code usando Planilhas:
  - AppSheet do Google - https://www.appsheet.com/
  - Bubble - https://bubble.io/
  - Glide - https://www.glideapps.com/

Arquivos de propriedades

- Um mapa importante é a tradicional classe Properties, que mapeia strings e é muito utilizada para a configuração de aplicações.

- A Properties possui, também, métodos para ler e gravar o mapeamento com base em um arquivo texto, facilitando muito a sua persistência.

```
user.name = angelica
user.mail = angelica@email.com
user.url = http://angelica.com.br
user.pass = 1##5932cd

book.title = Title
book.author = Author
book.edition = Edition
book.published = Published By
book.date = Published on
book.code = ISBN/ISNN
book.read = Read More about
```

Arquivos de propriedades

```
# Usando um dicionário para armazenar as propriedades
config = {
    "database.login": "scott",
    "database.password": "tiger",
    "database.url": "jdbc:mysql:/localhost/teste"
}

login = config.get("database.login")
password = config.get("database.password")
url = config.get("database.url")

# Exemplo de como poderia ser a conexão com um banco de dados em Python usando pymysql ou mysql.connector
import mysql.connector

# Estabelece a conexão usando os valores de login, password e url
connection = mysql.connector.connect(
    host="localhost",  # Retirando a parte do 'jdbc:mysql:/localhost/teste'
    database="teste",
    user=login,
    password=password
)
```

Arquivos de propriedades - gravação

```python
import configparser

# Cria um objeto ConfigParser, equivalente ao Properties em Java
config = configparser.ConfigParser()

# Adiciona as propriedades (chave-valor)
config['DEFAULT'] = {
    'database': 'localhost',
    'dbuser': 'mkyong',
    'dbpassword': 'password'
}

# Tenta gravar as propriedades no arquivo "config.ini"
with open('config.ini', 'w') as configfile:
    config.write(configfile)
```

- **`configparser.ConfigParser()`**: `ConfigParser` é uma biblioteca nativa do Python que lida com arquivos de configuração no formato `.ini`, semelhante ao arquivo `Properties` em Java.
- **`config['DEFAULT']`**: Define as propriedades no dicionário de configuração. No exemplo, as propriedades estão na seção `DEFAULT`, o que permite que sejam acessadas em qualquer parte do arquivo.
- **`config.write(configfile)`**: Grava o conteúdo do dicionário de configurações no arquivo `config.ini`.

Arquivos de propriedades - leitura

```python
import configparser

# Cria o objeto ConfigParser
config = configparser.ConfigParser()

# Lê o arquivo de propriedades (config.ini)
config.read('config.ini')

# Obtém os valores das propriedades
database = config['DEFAULT'].get('database')
dbuser = config['DEFAULT'].get('dbuser')
dbpassword = config['DEFAULT'].get('dbpassword')

# Imprime os valores
print(f"Database: {database}")
print(f"DB User: {dbuser}")
print(f"DB Password: {dbpassword}")
```

- **`config.read('config.ini')`**: Lê o arquivo de configuração `config.ini` (equivalente ao `config.properties` em Java).
- **`config['DEFAULT'].get('database')`**: Obtém o valor da chave `database` na seção `DEFAULT`. Em Python, usamos o método `get()` para buscar o valor associado a uma chave.
- **`print()`**: Imprime os valores das propriedades, equivalente ao `System.out.println()` do Java.

Manipulação de grandes arquivos

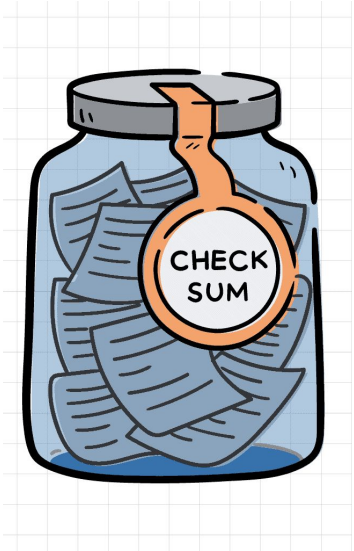
- **`wc -l big_file.csv`**: Conta o número de linhas no arquivo `big_file.csv`.
- **`head -n 3 big_file.csv`**: Exibe as primeiras 3 linhas do arquivo `big_file.csv`.
- **`tail -n 3 big_file.csv`**: Exibe as últimas 3 linhas do arquivo `big_file.csv`.
- **`cat big_file.csv | less`**: Exibe o conteúdo do arquivo `big_file.csv` paginado com o comando `less`, útil para navegar em arquivos muito grandes.
- **`cat big_file.csv | grep "52.2479 21.0191"`**: Procura e exibe as linhas no arquivo `big_file.csv` que contêm o texto "52.2479 21.0191".
- **`sed "s/ /;/g" teste.txt`**: Substitui todos os espaços ( ) por ponto e vírgula (`;`) no arquivo `teste.txt`. O comando `sed` é usado para substituições de texto.
- **`ls -l | tr -s ' ' | cut -d " " -f9`**: Lista os arquivos no diretório, compacta os espaços (`tr -s ' '` remove espaços extras) e depois usa `cut` para exibir apenas o nome dos arquivos (campo 9).

Manipulação de grandes arquivos

- **`ls -1`**: Lista os arquivos e diretórios, um por linha.
- **`awk '{ print $4 "," $5 }' teste.txt`**: Exibe o 4º e o 5º campo de cada linha do arquivo `teste.txt`, separando-os por vírgula. O `awk` é usado para manipulação de texto baseado em colunas.
- **`awk -F "," '{ print $4 "," $5 }' teste.csv`**: Igual ao comando anterior, mas define o delimitador como vírgula (`-F ","`) e aplica a um arquivo CSV.
- **`du -d1 | sort -n | cut -f2 | xargs du -hs`**: Calcula o uso de disco para subdiretórios no diretório atual, ordena pelo tamanho, e depois exibe de forma legível (`-h` para "human-readable").
- **`hexdump arquivo.txt`**: Exibe o conteúdo do arquivo `arquivo.txt` no formato hexadecimal.
- **`hexdump -C arquivo.txt`**: Exibe o conteúdo do arquivo `arquivo.txt` no formato hexadecimal e caracteres ASCII, com a saída formatada.
- **`hd arquivo.txt`**: Uma alternativa mais curta para o comando `hexdump`, que também exibe o arquivo em formato hexadecimal.

Checksum / Hash de grandes arquivos

- Checksum
  - cksum teste*.*
- MD5
  - md5sum teste*.* > md5.txt
  - md5sum -c md5.txt
- SHA1
  - sha1sum teste*.* > sha1.txt
  - sha1sum -c sha1.txt



Armazenando vários arquivos em um só arquivo (arquivamento)

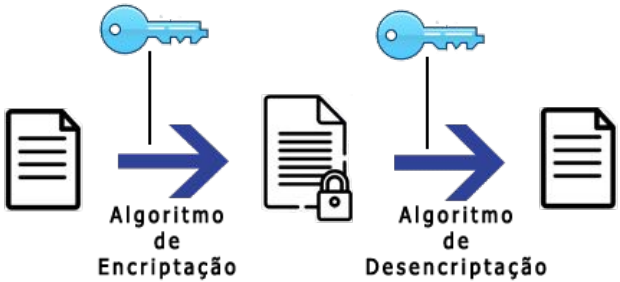
- tar
  - tar cfv teste.tar teste*.*
    - c – create
    - f – file
    - v - verbose

Compressão de arquivos

- **Sem perda**
  - zip, rar, 7z, gzip (um só arquivo), bzip2 (um só arquivo),...
  - zip teste.zip teste*.*
  - gzip teste.txt
  - bzip2 teste.txt
  - tar c teste*.* | gzip > teste.tar.gz
  - tar cfvz teste.tar.gz teste*.*
  - z - gzip
  - tar cfvj teste.tar.bz teste*.*
  - j – bzip2
- **Com perda**
  - jpg,mp3, mp4, ...

Encriptação de arquivos

- Encriptar:
  - gpg -c teste.txt
- Decriptar:
  - gpg teste.txt.gpg
- http://www.techrepublic.com/article/how-to-easily-encryptdecrypt-a-file-in-linux-with-gpg/
- http://irtfweb.ifa.hawaii.edu/~lockhart/gpg/
- https://help.ubuntu.com/community/GnuPrivacyGuardHowto



Leitura de arquivo ZIP

```python
import zipfile

# Caminho do arquivo ZIP
zip_file_path = "./teste.zip"

# Abre o arquivo ZIP para leitura
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    # Lista os arquivos dentro do ZIP
    for file_name in zip_ref.namelist():
        # Abre o arquivo dentro do ZIP
        with zip_ref.open(file_name) as file:
            # Lê o conteúdo do arquivo
            for line in file:
                # Converte de bytes para string e remove quebras de
linha
                print(line.decode('utf-8').strip())
```

- `zipfile.ZipFile()`: Abre o arquivo ZIP para leitura.
- `zip_ref.namelist()`: Lista os nomes dos arquivos dentro do arquivo ZIP.
- `zip_ref.open(file_name)`: Abre um arquivo específico dentro do ZIP para leitura.
- `file.read()` **ou iteração linha por linha**: Lê o conteúdo do arquivo compactado. Em Python, as linhas são lidas como bytes, então usamos `.decode('utf-8')` para convertê-las em strings.
- `.strip()`: Remove qualquer quebra de linha ou espaços adicionais.

Leitura de arquivo ZIP

```
import zipfile

# Caminho do arquivo ZIP
zip_file_path = "./tb1.zip"

# Abre o arquivo ZIP
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    # Pega o nome do primeiro arquivo dentro do ZIP
    file_name = zip_ref.namelist()[0]

    # Abre o arquivo dentro do ZIP
    with zip_ref.open(file_name) as file:
        # Usa o scanner (como o Scanner no Java, lendo linha por linha)
        for line in file:
            print(line.decode('utf-8').strip())  # Converte bytes para
string e remove quebras de linha
```

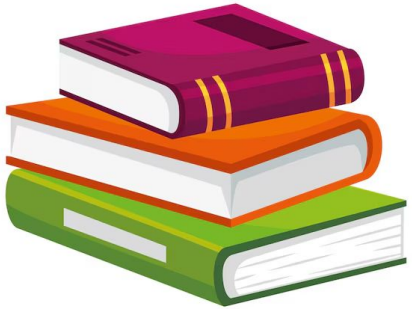
- `zipfile.ZipFile(zip_file_path, 'r')`: Abre o arquivo ZIP para leitura.
- `zip_ref.namelist()[0]`: Obtém o nome do primeiro arquivo dentro do ZIP.
- `zip_ref.open(file_name)`: Abre o arquivo dentro do ZIP para leitura.
- `for line in file:`: Lê o conteúdo do arquivo dentro do ZIP linha por linha.
- `.decode('utf-8').strip()`: Converte cada linha (que é lida como bytes) para uma string e remove quebras de linha ou espaços em branco adicionais.

Bibliografia Básica

- SADALAGE, P. J. E FOWLER, M. NoSQL Essencial. Editora Novatec, São Paulo, 2013.
- REDMOND, E.; WILSON, J. R. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. 1ª edição, 2012. The Pragmatic Programmers.
- ULLMAN, J.D.; WIDOW, J. First Course in Database Systems. 3a edição, 2007. Prentice Hall.
- HAMBRICK, G. et al. Persistence in the Enterprise: A Guide to Persistence Technologies; 1ª edição, 2008. IBM Press.
- ELMASRI, R.; NAVATHE, S. B. Sistemas de banco de dados. 4ª edicao, 2009. Pearson/Addison-Wesley.

Bibliografia Complementar

- WHITE, Tom. Hadoop: the definitive guide. California: O´Reilly, 2009. xix, 501 p. ISBN 9780596521974 (broch.).
- AMBLER, S.W., SADALAGE, P.J. Refactoring Databases: Evolutionary Database Design. 1a edição, 2011. Addison Wesley.
- SILBERSCHATZ, A.; SUDARSHAN, S. Sistema de banco de dados. 2006. Campus.
- LYNN, B. Use a cabeça! SQL. 1ª edição, 2008. ALTA BOOKS.
- SMITH, Ben. JSON básico: conheça o formato de dados preferido da web. São Paulo: Novatec, 2015. 400 p. ISBN 9788575224366 (broch.).
- HITZLER, P., KRÖTZSCH, M., and RUDOLPH, S. (2009). Foundations of Semantic Web Technologies. Chapman & Hall/CRC.
- ANTONIOU, G. and HARMELEN, F. (2008). A Semantic Web Primer. Second Edition,Cambridge, MIT Press, Massachusetts.
- HEATH, T. and BIZER, C. (2011). Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool, 1st edition.



Obrigado!

Dúvidas?

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br